modular micros

martin research

# MIKE 3

# MANUAL

MM811

*with update materials*
*for*
*Model AT814 DEBUG System....*

Preserved by
aconit.org

modular micros | martin research

*April, 1976* *REV. C*


| SECTION | | DESCRIPTION | PAGES |
|---|---|---|---|
| 1. | INTRODUCTION | *Modular Microcomputers...from the People Who Wrote the Book!* | 2 |
| | | DS800--Modular Micros, an Introduction | 2 |
| 2. | CONSTRUCTION NOTES | KN800--Construction Notes | 2 |
| 3. | MICROCOMPUTERS | *Modular Micros!* | 1 |
| | | DS801--Introduction to Microcomputers | 1 |
| | | DS2204--8080A Microprocessor | 30 |
| 4. | THE SYSTEM | DS811/Sec. 4--The *MIKE 3* System | 2 |
| 5. | OPERATING | OP811--Operating Instructions | 11 |
| 6. | CPU | DS471--8080 Central Processing Unit | 23 |
| | | DS431--8080 CPU* | 5* |
| 7. | MEMORY | DS811/Sec. 7--Memory in the *MIKE 3* | 1 |
| | | DS405--4K RAM *(schematic, pp. 4-6 only)* | 3 |
| | | DS423--PROM/RAM board *(pictorial--Fig. 3.1.1--and schematics--pp. 4-6--only)* | 4 |
| 8. | I/O | DS811/Sec. 8--Inputs and Outputs in the *MIKE 3* | 1 |
| | | DS420--Console Board *(schematics, pp. 4-6 only)* | 2 |
| | | DS9001--Universal I/O Kit *(schematics, pp. 5 and 7 only)* | 2 |
| | | *ICs Interface Keyboard to Microprocessor (reprint from* Electronics, *3/16/75)* | 2 |
| 9. | POWER | DS811/Sec. 9--Power Supply for the *MIKE 3* | 1 |
| 10. | ACCESSORIES | DS11000--110 Character-Per-Second Printer | 1 |
| 11. | SERVICE | Warranty Policy; Return Policy | 1 |
| 12. | SOFTWARE | SP2220-16/17--*MIKE 3 Monitor* Program | 14 |
| | | SP2220-15--*MIKE 3 Demo PROM* | 3 |


*MIKE 2 MANUAL* This 176-page paperback is included with each *MIKE 3 Manual*. It includes full information on the Model 405 4K RAM, 420 Console, and 423 PROM/RAM boards, as well as on the 8008-based *MIKE 2* Microcomputer.

* [*The 431 CPU data sheet is provided in early Manuals; replaced by 471.*]

April, 1976 REV. A

In addition to the information provided with the basic MIKE 3 Manual, the following information is supplied for users of the MIKE 3 DEBUG System, Model AT814. This MIKE 3 microcomputer includes a versatile DEBUG PROM, for single-stepping through programs (plus other features), and a Model 472 DEBUG board, with a 32-Channel Scope Display.

For details, see the specific data sheets mentioned below.

| SECTION | DESCRIPTION | Pages |
|---|---|---|
| 5. OPERATING | OP814--Operating Instructions for the DEBUG System | 14 |
| 6. CPU | DS472--Model 472 DEBUG Board | 14 |
| 12. SOFTWARE | SP2220-24C--MIKE 3 DEBUG PROM | 1* |

* The software listing for the DEBUG PROM is provided only to purchasers.

Note: Purchasers of earlier MIKE 3 microcomputers can upgrade their systems to the DEBUG version. Required prerequisites:

Model 471 CPU board, with 8080A;
Model 420 Console board (keyboard and LED digits);
Model 423 PROM/RAM board, with MIKE 3 Monitor, octal version, PROM 2200-17, D revision or later. (Users holding earlier PROMs may trade them in for reprogramming at no charge.)

Required additions to support DEBUG features:

1 each AT472-2 DEBUG board, with 20-conductor connectors, $89.00
1 each P/N 1001, 50-pin transition connector, for installation on system ribbon cable, $6.00
1 each MIKE 3 DEBUG PROM, P/N 2200-24C, for 423 board, $32.00

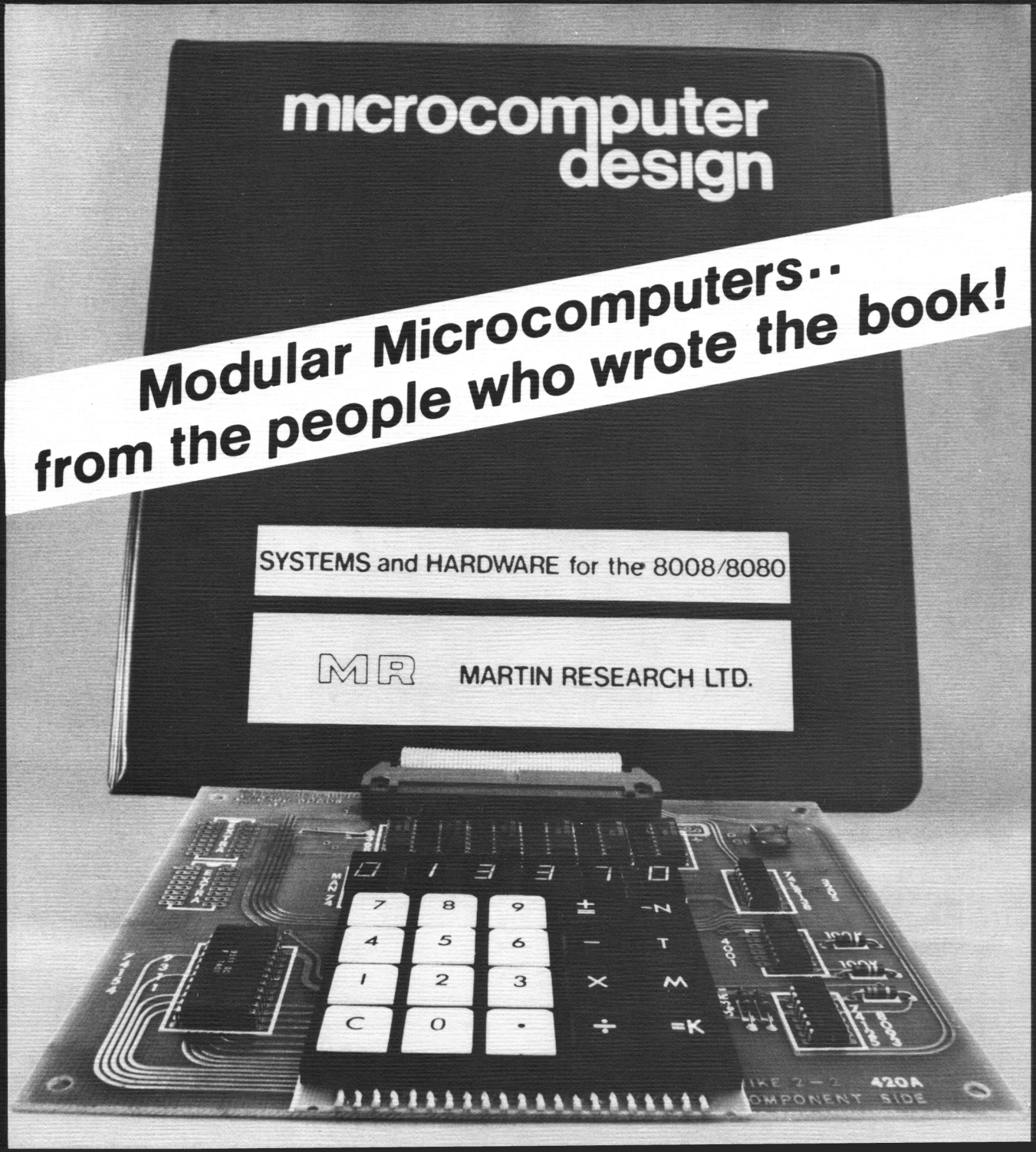
Total list price, $127.00. Order as assembly, Catalog AT816, $119.00.

U.S. prices, subject to change without notice. Standard terms and conditions.

The AT472-2 is assembled and tested.

# martin research

3336 Commercial Avenue Northbrook, IL 60062
(312)498-5060



# modular micros!

3336 Commercial
Northbrook, IL
(312) 498-5060

We are proud to announce the *modular micros* from Martin Research. This versatile series of printed circuit modules can be used to assemble a variety of microcomputer systems, from the simple to the complex.

✓ *Not* just a set of parts, with the wirewrapping left to you....but complete with commercial-grade printed circuit boards, made by top PC houses, with plated-through holes.

✓ The system architecture has been designed for optimum flexibility. The *modular micros* use a carefully-designed bus structure, permitting any board to be inserted in any position on the bus. A flat cable and snap-on fifty-pin connectors make for easy expansion.

✓ The universal bus structure is compatible with all standard 8-bit CPUs: a major advantage for the user who wishes to prototype systems with competitive processors. Memory and accessory boards remain compatible when the CPU is changed. The 8080-based *MIKE 3* system, and the 8008-based *MIKE 2*, are both being delivered. Other CPUs are planned, including the 6800/6501 series.

✓ Our 4K static RAM board uses RAM chips fast enough--450 ns--to avoid making the CPU wait for memory access in most applications. Now in stock, this board sells for less than many competitive memory boards with far slower specifications.

✓ Superior systems design is featured in the *modular micro* series. Program entry is performed on a calculator-style keyboard, and six bright LED digits display data and memory addresses. A Monitor program in a PROM makes program entry easy. This is a significant advance over console designs with cumbersome arrays of toggle switches and lamps.

✓ With microcomputer systems starting at under $400, the *modular micros* offer a real opportunity to the manufacturer prototyping his own μP-based designs; to the educator introducing his students to this new technology; and to the engineer, for his own personal use in keeping up with the state of the art.

*FOR DETAILS, ASK FOR THE COMPLETE* MODULAR MICRO *CATALOG!*

*About the book*....published in 1974, *Microcomputer Design* is an industry standard for the engineer working with microprocessors. For a chapter outline and pricing information, contact Martin Research.

*Now: available from Semiconductor Specialists!* Chicago, (312) 279-1000; Los Angeles, (213) 641-3323; Detroit, (313) 478-2700; Minneapolis, (612) 854-8841; Kansas City, (816) 452-3900; Dallas, (214) 358-5211; St. Louis, (314) 731-2400; Indianapolis, (317) 243-8271; Dayton, (513) 278-9455; Pittsburgh, (412) 781-8120; Milwaukee, (414) 257-1330; Canada (nr. Toronto), (416) 678-1444; England, (W. Drayton) 46415; Scotland, (Livingston) 31967; Germany (Frankfurt/Main Sud), (0611) 622033.

**modular micros** ⇄ **martin research**

# 1. Modular Micros: an Introduction

Dear Customer:

We hope that this *modular micro* Manual is useful to you.

In order to keep this Manual current, Martin Research maintains a special Manual update mailing list. Already on this list are those customers who purchased this Manual, or a *modular micro* system, directly from Martin Research. Initially, all such customers automatically receive a one-year's subscription to these updates, ending December, 1976. Until then, supplements are being sent without charge to fill in incomplete sections in the Manual.

If you wish to confirm your subscription to the Manual, please use the coupon provided below. Also, please keep us current on your address. Martin Research cannot be responsible for delivery of updates unless notified of any address change, preferably one month in advance. Write to the attention of the MANUAL UPDATE DEPARTMENT.

---

Send to: Martin Research, 3336 Commercial Ave., Northbrook, Illinois 60062 USA

ATTN: MANUAL UPDATE DEPARTMENT *Model No.*_____

_____ Please make sure my subscription is entered. *Date Purchased*_____

_____ Please enter a subscription; *modular micro* purchased from_____

_____ Change of address, effective on_____

NAME_____

ADDRESS_____

_____

_____ZIP CODE_____

REV. A

Martin Research hereby solicits your comments and criticisms regarding this Manual. Users' ideas for hardware and software for the *modular micro* series will gladly be received. Subject to evaluation and editing, those judged most useful will be distributed to other Manual holders.

This sheet may be used for your comments, and returned to Martin Research.

© 1975, Martin Research, Northbrook, IL 60062 USA. All rights reserved.

| 5 | 4 | 3 | 2 | Rev | ECO | By | Date |
|---|---|---|---|---|---|---|---|
| A | A | A | A | A | 016 | KB | 12/4/75 |

# 2. Construction: Notes for the Kit-Builder

Circuit boards and systems in the *modular micro* series which are sold in assembled and tested form (prefix *AT*) are shipped ready for operation. Purchasing tested boards is the best way to obtain a debugged system with a minimum of labor and testing.

*Modular micro* boards and systems are also sold as kits (prefix *KT*). To date kits have usually been shipped partially assembled. Beginning with the introduction of the 804 system, all *modular micro* boards are sold with sockets for every IC included. The exception is the Model 420 Console board, where for appearance's sake, the ICs are soldered directly into the board.

When a *modular micro* board is shipped as a collection of parts plus PC board, supplemental instructions are provided.

The remainder of this sheet describes certain construction details common to components in the *modular micro* series.

## SEC. 1 WARNING! STATIC DESTROYS MOS!

Some critical microcomputer components are manufactured using *MOS* (metal oxide semiconductor) technology. These devices can be destroyed by static electricity. Examples include the 8080, 6800, and 8008 microprocessors; 1702 PROMs; and 2102 and 2112 RAMs.

A MOS device should either be stored in black conductive foam or a metal IC container, or be plugged into an IC socket which has been soldered into a *modular micro* board. Avoid unnecessary handling of unprotected MOS ICs. Use their ceramic or plastic packages to pick them up, rather than their metal terminals. Avoid wearing nylon clothing, or walking across carpeted rooms, especially under dry atmospheric conditions. Either work on a grounded bench, or contact a grounded surface (water pipe or electrical conduit) by hand to discharge static buildup *before* handling MOS devices. Remove power from a circuit before inserting or removing a MOS IC.

## SEC. 2 ASSEMBLY OF THE *MODULAR MICRO* STACK

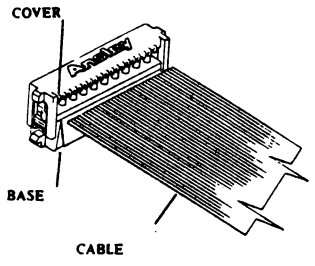
The standard *modular micro* board measures 5.5 by 7.0 inches (140 mm by 178 mm), and is punched with holes at the four corners to clear #6-32 threads. The boards are assembled as a stack, with the boards separated by 5/8-inch hexagonal spacers. Each spacer has one female and one male end, with #6-32 threads; spacers may conveniently be tightened by hand or with a ¼-inch hex wrench.

Each board is equipped with a fifty-pin PC board *header,* plugging into a *transition connector* on a fifty-wire cable. Boards may be added to the system by attaching transition connectors to the cable, mounting additional hex spacers on the stack, and plugging in the new components. During testing and prototyping, only one or two spacers are necessary to separate boards--rather than using all four columns. This facilitates rapid insertion and removal of boards from the stack.

## SEC. 3 FIFTY-WIRE CABLE SYSTEM

The fifty-pin connection system used in the *modular micro* series offers maximum flexibility to the user. Enough cable is provided with each system for a reasonable amount of expansion; or, cable may be purchased from Martin Research. When the boards are to be assembled in a stack, using 5/8" spacers, the recommended distance (center to center) between transition connectors is 3/4" (1.9 cm). Longer spacing may require the cable to bend too sharply, putting strain on the connections to the boards.

The user can add a transition connector to the cable with an ordinary bench vise. The connectors supplied by Martin Research are recommended, since some brands require special assembly equipment, and once attached cannot be removed. The transition connector comes in two parts, a *cable cover* and *base.* Assembly steps: with the cable cover placed loosely over the base, slide the cable into its desired position. Now hold the cable assembly up to a light and make sure each of the fifty wires is directly above one of the fifty V-notched teeth on the connector base. Ensure also that the transition connector is at right angles to the run of the cable. Press the cable cover and base together by hand, clamping the cable. Using a vise, *gently and evenly* apply pressure to lock the teeth into the wire. Do this in gradual steps until the cable cover locks onto the base. The ears at the ends of the cable cover might not automatically lock onto the tabs on the base, even when the connector is securely clamped. Try locking the two connector sections by hand before applying still more vise pressure, which could damage the connector.




The Ansley brand of transition connectors may be removed from the cable and repositioned. If the work is done carefully, both connector and cable will still be serviceable.

©1975, Martin Research. Pp. 1-2, Rev. A, Nov. 1975. All rights reserved.

# 3. Microcomputers: an Introduction

# modular micros!

We are proud to announce the *modular micros* from Martin Research. These versatile printed circuit modules can be configured to produce a variety of microcomputer systems, from the simple to the complex.

## BEST VALUE !

*Not* just a collection of parts, with the wirewrapping left to you . . . . but complete with commercial-grade printed circuit boards, made by top PC houses, with plated-through holes. Minimizes construction time, and greatly reduces troubleshooting.

## MOST ADAPTABLE!

Our universal bus structure is compatible with all standard eight-bit microprocessors! This is a major advantage for the user who wishes to upgrade his system in the future. And, it is ideal for the OEM building prototypes with competing processors. Memory and accessory boards remain compatible when the CPU is changed.

## MOST FLEXIBLE!

The system architecture, designed for optimum flexibility, allows any board to be inserted in any position on the bus. A flat cable and snap-on fifty-pin connectors make for easy expansion.

## SUPERIOR SYSTEMS DESIGN!

Programs are entered on a calculator-style keyboard, and six bright LED digits display data and memory addresses. A Monitor program in a PROM makes program entry easy. This is a significant advance over console designs with cumbersome arrays of toggle switches and lamps.

## FAST MEMORY AVAILABLE!

Our 4K static RAM board uses RAM chips fast enough--450 ns--to avoid making the CPU wait for memory access in most applications. Now in stock, this board is price-competitive with others' memory boards with far slower specifications. Warning: some manufacturers using slow RAM do not inform the buyer of this fact. The result is that the CPU must wait for memory, resulting in a computer much slower than the specs might imply.

## BEST FOR YOUR APPLICATION!

With microcomputer systems starting at under $400, the *modular micros* are ideal for:

- The computer enthusiast, in his home;
- The professional engineer or programmer, to keep up with the state of the art;
- The manufacturer, for prototyping microcomputer systems, or for incorporation in his product.

*(OEMs: write for further information.)*

REV. A

**modular micros** **martin research**

This Manual includes complete circuit descriptions for all microcomputer components, with circuit diagrams and software listings. As such, it provides a good introduction to microprocessor technology.

For the reader new to microprocessors, a recommended start is to review the manufacturer's data sheet on the microprocessor, reprinted in this section of the Manual. Then, read the circuit description of the CPU board, in Section 6. Finally, look over the Operating Instructions, Section 5 (and the Software description in Section 12).

The microcomputer designer wishing a more comprehensive treatment may wish to study the book published by Martin Research, *Microcomputer Design*.

In any case, the best introduction is hands-on experience with a functioning microcomputer. This is best provided by working with a *modular micro* system, understanding its operations, and gaining facility in programming.

*This data sheet reprints literature published by Intel Corporation. The 8080A, part number 2204, is the heart of the MIKE 3 microcomputer from Martin Research.*

## ARCHITECTURE OF THE 8080 CPU

The 8080 CPU consists of the following functional units:

- Register array and address logic
- Arithmetic and logic unit (ALU)
- Instruction register and control section
- Bi-directional, 3-state data bus buffer

### Registers:

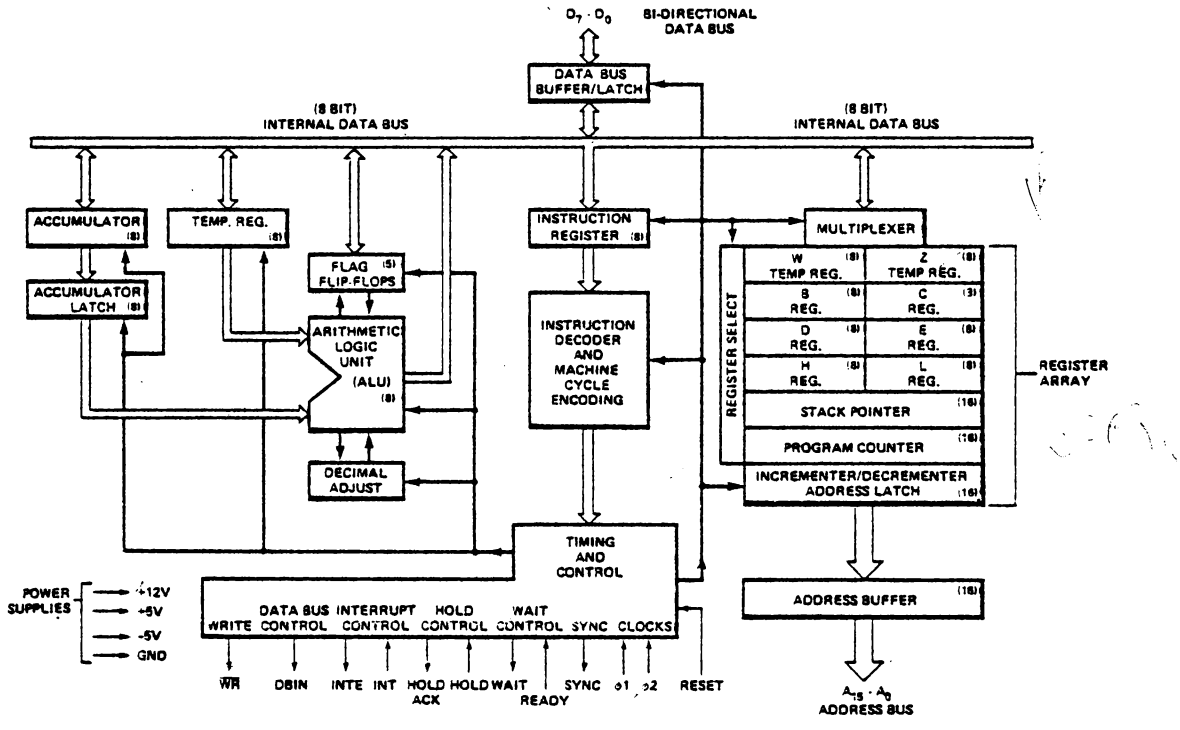
The register section consists of a static RAM array organized into six 16-bit registers:

- Program counter (PC)
- Stack pointer (SP)
- Six 8-bit general purpose registers arranged in pairs, referred to as B,C; D,E; and H,L
- A temporary register pair called W,Z

The program counter maintains the memory address of the current program instruction and is incremented automatically during every instruction fetch. The stack pointer maintains the address of the next available stack location in memory. The stack pointer can be initialized to use any portion of read-write memory as a stack. The stack pointer is decremented when data is "pushed" onto the stack and incremented when data is "popped" off the stack (i.e., the stack grows "downward").

The six general purpose registers can be used either as single registers (8-bit) or as register pairs (16-bit). The temporary register pair, W,Z, is not program addressable and is only used for the internal execution of instructions.

Eight-bit data bytes can be transferred between the internal bus and the register array via the register-select multiplexer. Sixteen-bit transfers can proceed between the register array and the address latch or the incrementer/decrementer circuit. The address latch receives data from any of the three register pairs and drives the 16 address output buffers ($A_0$-$A_{15}$), as well as the incrementer/decrementer circuit. The incrementer/decrementer circuit receives data from the address latch and sends it to the register array. The 16-bit data can be incremented or decremented or simply transferred between registers.




8080 CPU Functional Block Diagram

modular micros martin research

### Arithmetic and Logic Unit (ALU):

The ALU contains the following registers:

- An 8-bit accumulator
- An 8-bit temporary accumulator (ACT)
- A 5-bit flag register: zero, carry, sign, parity and auxiliary carry
- An 8-bit temporary register (TMP)

Arithmetic, logical and rotate operations are performed in the ALU. The ALU is fed by the temporary register (TMP) and the temporary accumulator (ACT) and carry flip-flop. The result of the operation can be transferred to the internal bus or to the accumulator; the ALU also feeds the flag register.

The temporary register (TMP) receives information from the internal bus and can send all or portions of it to the ALU, the flag register and the internal bus.

The accumulator (ACC) can be loaded from the ALU and the internal bus and can transfer data to the temporary accumulator (ACT) and the internal bus. The contents of the accumulator (ACC) and the auxiliary carry flip-flop can be tested for decimal correction during the execution of the DAA instruction.

### Instruction Register and Control:

During an instruction fetch, the first byte of an instruction (containing the OP code) is transferred from the internal bus to the 8-bit instruction register.

The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, combined with various timing signals, provides the control signals for the register array, ALU and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and cycle timing signals.

### Data Bus Buffer:

This 8-bit bidirectional 3-state buffer is used to isolate the CPU's internal bus from the external data bus ($D_0$ through $D_7$). In the output mode, the internal bus content is loaded into an 8-bit latch that, in turn, drives the data bus output buffers. The output buffers are switched off during input or non-transfer operations.

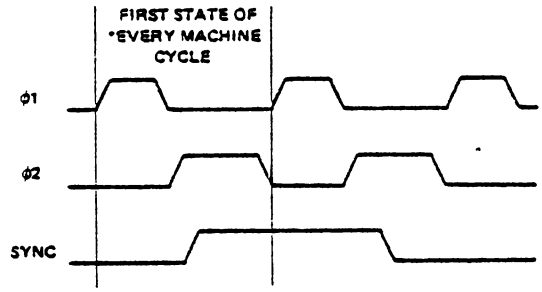
During the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state $T_3$—described later.

## THE PROCESSOR CYCLE

An instruction cycle is defined as the time required to fetch and execute an instruction. During the fetch, a selected instruction (one, two or three bytes) is extracted from memory and deposited in the CPU's instruction register. During the execution phase, the instruction is decoded and translated into specific processing activities.

Every instruction cycle consists of one, two, three, four or five machine cycles. A machine cycle is required each time the CPU accesses memory or an I/O port. The fetch portion of an instruction cycle requires one machine cycle for each byte to be fetched. The duration of the execution portion of the instruction cycle depends on the kind of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to/from memory or I/O devices. The DAD instruction is an exception in that it requires two additional machine cycles to complete an internal register-pair add.

Each machine cycle consists of three, four or five states. A state is the smallest unit of processing activity and is defined as the interval between two successive positive-going transitions of the $\phi_1$ driven clock pulse. The 8080 is driven by a two-phase clock oscillator. All processing activities are referred to the period of this clock. The two non-overlapping clock pulses, labeled $\phi_1$ and $\phi_2$, are furnished by external circuitry. It is the $\phi_1$ clock pulse which divides each machine cycle into states. Timing logic within the 8080 uses the clock inputs to produce a SYNC pulse, which identifies the beginning of every machine cycle. The SYNC pulse is triggered by the low-to-high transition of $\phi_2$.




*SYNC DOES NOT OCCUR IN THE SECOND AND THIRD MACHINE CYCLES OF A DAD INSTRUCTION SINCE THESE MACHINE CYCLES ARE USED FOR AN INTERNAL REGISTER-PAIR ADD.

$\phi_1, \phi_2$ And SYNC Timing

There are three exceptions to the defined duration of a state. They are the WAIT state, the hold (HLDA) state and the halt (HLTA) state, described later in this chapter. Because the WAIT, the HLDA, and the HLTA states depend upon external events, they are by their nature of indeterminate length. Even these exceptional states, however, must

*The information in this data sheet has been reprinted with the permission of the copyright holder, Intel Corporation. Copyright ©1975, Intel Corp.*

modular micros → martin research →

be synchronized with the pulses of the driving clock. Thus, the duration of all states are integral multiples of the clock period.

To summarize then, each clock period marks a state; three to five states constitute a machine cycle; and one to five machine cycles comprise an instruction cycle. A full instruction cycle requires anywhere from four to eighteen states for its completion, depending on the kind of instruction involved.

## Machine Cycle Identification:

With the exception of the DAD instruction, there is just one consideration that determines how many machine cycles are required in any given instruction cycle: the number of times that the processor must reference a memory address or an addressable peripheral device, in order to fetch and execute the instruction. Like many processors, the 8080 is so constructed that it can transmit only one address per machine cycle. Thus, if the fetch and execution of an instruction requires two memory references, then the instruction cycle associated with that instruction consists of two machine cycles. If five such references are called for, then the instruction cycle contains five machine cycles.

Every instruction cycle has at least one reference to memory, during which the instruction is fetched. An instruction cycle must always have a fetch, even if the execution of the instruction requires no further references to memory. The first machine cycle in every instruction cycle is therefore a FETCH. Beyond that, there are no fast rules. It depends on the kind of instruction that is fetched.

Consider some examples. The add-register (ADD r) instruction is an instruction that requires only a single machine cycle (FETCH) for its completion. In this one-byte instruction, the contents of one of the CPU's six general purpose registers is added to the existing contents of the accumulator. Since all the information necessary to execute the command is contained in the eight bits of the instruction code, only one memory reference is necessary. Three states are used to extract the instruction from memory, and one additional state is used to accomplish the desired addition. The entire instruction cycle thus requires only one machine cycle that consists of four states, or four periods of the external clock.

Suppose now, however, that we wish to add the contents of a specific memory location to the existing contents of the accumulator (ADD M). Although this is quite similar in principle to the example just cited, several additional steps will be used. An extra machine cycle will be used, in order to address the desired memory location.

The actual sequence is as follows. First the processor extracts from memory the one-byte instruction word addressed by its program counter. This takes three states. The eight-bit instruction word obtained during the FETCH machine cycle is deposited in the CPU's instruction register and used to direct activities during the remainder of the instruction cycle. Next, the processor sends out, as an address, the contents of its H and L registers. The eight-bit data word returned during this MEMORY READ machine cycle is placed in a temporary register inside the 8080 CPU. By now three more clock periods (states) have elapsed. In the seventh and final state, the contents of the temporary register are added to those of the accumulator. Two machine cycles, consisting of seven states in all, complete the "ADD M" instruction cycle.

At the opposite extreme is the save H and L registers (SHLD) instruction, which requires five machine cycles. During an "SHLD" instruction cycle, the contents of the processor's H and L registers are deposited in two sequentially adjacent memory locations; the destination is indicated by two address bytes which are stored in the two memory locations immediately following the operation code byte. The following sequence of events occurs:

(1) A FETCH machine cycle, consisting of four states. During the first three states of this machine cycle, the processor fetches the instruction indicated by its program counter. The program counter is then incremented. The fourth state is used for internal instruction decoding.

(2) A MEMORY READ machine cycle, consisting of three states. During this machine cycle, the byte indicated by the program counter is read from memory and placed in the processor's Z register. The program counter is incremented again.

(3) Another MEMORY READ machine cycle, consisting of three states, in which the byte indicated by the processor's program counter is read from memory and placed in the W register. The program counter is incremented, in anticipation of the next instruction fetch.

(4) A MEMORY WRITE machine cycle, of three states, in which the contents of the L register are transferred to the memory location pointed to by the present contents of the W and Z registers. The state following the transfer is used to increment the W,Z register pair so that it indicates the next memory location to receive data.

(5) A MEMORY WRITE machine cycle, of three states, in which the contents of the H register are transferred to the new memory location pointed to by the W,Z register pair.

In summary, the "SHLD" instruction cycle contains five machine cycles and takes 16 states to execute.

Most instructions fall somewhere between the extremes typified by the "ADD r" and the "SHLD" instructions. The input (INP) and the output (OUT) instructions, for example, require three machine cycles: a FETCH, to obtain the instruction; a MEMORY READ, to obtain the address of the object peripheral; and an INPUT or an OUTPUT machine cycle, to complete the transfer.

modular micros → martin research →

While no one instruction cycle will consist of more then five machine cycles, the following ten different types of machine cycles may occur within an instruction cycle:

(1) FETCH (M1)
(2) MEMORY READ
(3) MEMORY WRITE
(4) STACK READ
(5) STACK WRITE
(6) INPUT
(7) OUTPUT
(8) INTERRUPT
(9) HALT
(10) HALT • INTERRUPT

The machine cycles that actually do occur in a particular instruction cycle depend upon the kind of instruction, with the overriding stipulation that the first machine cycle in any instruction cycle is always a FETCH.

The processor identifies the machine cycle in progress by transmitting an eight-bit status word during the first state of every machine cycle. Updated status information is presented on the 8080's data lines ($D_0$-$D_7$), during the SYNC interval. This data should be saved in latches, and used to develop control signals for external circuitry. Table 2-1 shows how the positive-true status information is distributed on the processor's data bus.

Status signals are provided principally for the control of external circuitry. Simplicity of interface, rather than machine cycle identification, dictates the logical definition of individual status bits. You will therefore observe that certain processor machine cycles are uniquely identified by a single status bit, but that others are not. The $M_1$ status bit ($D_6$), for example, unambiguously identifies a FETCH machine cycle. A STACK READ, on the other hand, is indicated by the coincidence of STACK and MEMR signals. Machine cycle identification data is also valuable in the test and de-bugging phases of system development. Table 2-1 lists the status bit outputs for each type of machine cycle.

## State Transition Sequence:

Every machine cycle within an instruction cycle consists of three to five active states (referred to as $T_1$, $T_2$, $T_3$, $T_4$, $T_5$ or $T_W$). The actual number of states depends upon the instruction being executed, and on the particular machine cycle within the greater instruction cycle. The state transition diagram in Figure 2-4 shows how the 8080 proceeds from state to state in the course of a machine cycle. The diagram also shows how the READY, HOLD, and INTERRUPT lines are sampled during the machine cycle, and how the conditions on these lines may modify the basic transition sequence. In the present discussion, we are concerned only with the basic sequence and with the READY function. The HOLD and INTERRUPT functions will be discussed later.

The 8080 CPU does not directly indicate its internal state by transmitting a "state control" output during each state; instead, the 8080 supplies direct control output (INTE, HLDA, DBIN, $\overline{WR}$ and WAIT) for use by external circuitry.

Recall that the 8080 passes through at least three states in every machine cycle, with each state defined by successive low-to-high transitions of the $\phi_1$ clock. Figure 2-5 shows the timing relationships in a typical FETCH machine cycle. Events that occur in each state are referenced to transitions of the $\phi_1$ and $\phi_2$ clock pulses.

The SYNC signal identifies the first state ($T_1$) in every machine cycle. As shown in Figure 2-5, the SYNC signal is related to the leading edge of the $\phi_2$ clock. There is a delay ($t_{DC}$) between the low-to-high transition of $\phi_2$ and the positive-going edge of the SYNC pulse. There also is a corresponding delay (also $t_{DC}$) between the next $\phi_2$ pulse and the falling edge of the SYNC signal. Status information is displayed on $D_0$-$D_7$ during the same $\phi_2$ to $\phi_2$ interval. Switching of the status signals is likewise controlled by $\phi_2$.

The rising edge of $\phi_2$ during $T_1$ also loads the processor's address lines ($A_0$-A15). These lines become stable within a brief delay ($t_{DA}$) of the $\phi_2$ clocking pulse, and they remain stable until the first $\phi_2$ pulse after state $T_3$. This gives the processor ample time to read the data returned from memory.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a WAIT. This it does by pulling the processor's READY line low, prior to the "Ready set-up" interval ($t_{RS}$) which occurs during the $\phi_2$ pulse within state $T_2$ or $T_W$. As long as the READY line remains low, the processor will idle, giving the memory time to respond to the addressed data request. Refer to Figure 2-5.

The processor responds to a wait request by entering an alternative state ($T_W$) at the end of $T_2$, rather than proceeding directly to the $T_3$ state. Entry into the $T_W$ state is indicated by a WAIT signal from the processor, acknowledging the memory's request. A low-to-high transition on the WAIT line is triggered by the rising edge of the $\phi_1$ clock and occurs within a brief delay ($t_{DC}$) of the actual entry into the $T_W$ state.

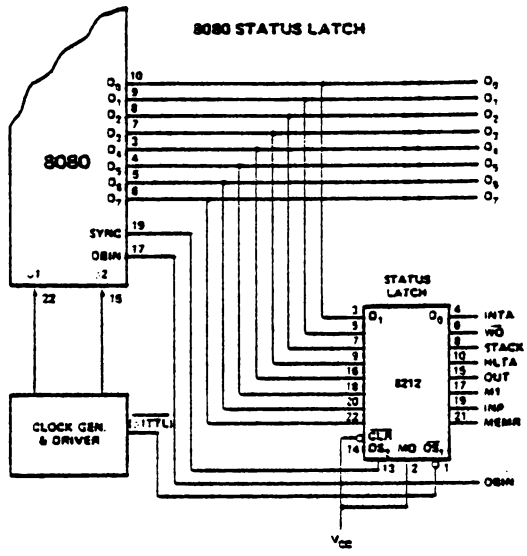
A wait period may be of indefinite duration. The processor remains in the waiting condition until its READY line again goes high. A READY indication must precede the falling edge of the $\phi_2$ clock by a specified interval ($t_{RS}$), in order to guarantee an exit from the $T_W$ state. The cycle may then proceed, beginning with the rising edge of the next $\phi_1$ clock. A WAIT interval will therefore consist of an integral number of $T_W$ states and will always be a multiple of the clock period.

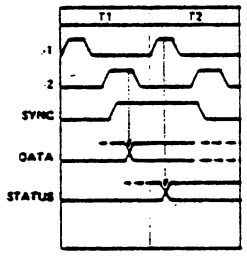
Instructions for the 8080 require from one to five machine cycles for complete execution. The 8080 sends out 8 bit of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

**STATUS INFORMATION DEFINITION**

| Symbols | Data Bus Bit | Definition |
|---|---|---|
| INTA* | $D_0$ | Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active. |
| $\overline{WO}$ | $D_1$ | Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed. |
| STACK | $D_2$ | Indicates that the address bus holds the pushdown stack address from the Stack Pointer. |
| HLTA | $D_3$ | Acknowledge signal for HALT instruction. |
| OUT | $D_4$ | Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active. |
| $M_1$ | $D_5$ | Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction. |
| INP* | $D_6$ | Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active. |
| MEMR* | $D_7$ | Designates that the data bus will be used for memory read data. |

*These three status bits can be used to control the flow of data onto the 8080 data bus.




**STATUS WORD CHART**

TYPE OF MACHINE CYCLE

| DATA BUS BIT | STATUS INFORMATION | INSTRUCTION FETCH | MEMORY READ | MEMORY WRITE | STACK READ | STACK WRITE | INPUT READ | OUTPUT WRITE | INTERRUPT ACKNOWLEDGE | HALT ACKNOWLEDGE | INTERRUPT ACKNOWLEDGE WHILE HALT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) | (10) |
| $D_0$ | INTA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $D_1$ | $\overline{WO}$ | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| $D_2$ | STACK | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| $D_3$ | HLTA | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| $D_4$ | OUT | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| $D_5$ | $M_1$ | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| $D_6$ | INP | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| $D_7$ | MEMR | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

(N) STATUS WORD

Table 2-1. 8080 Status Bit Definitions

modular micros martin research

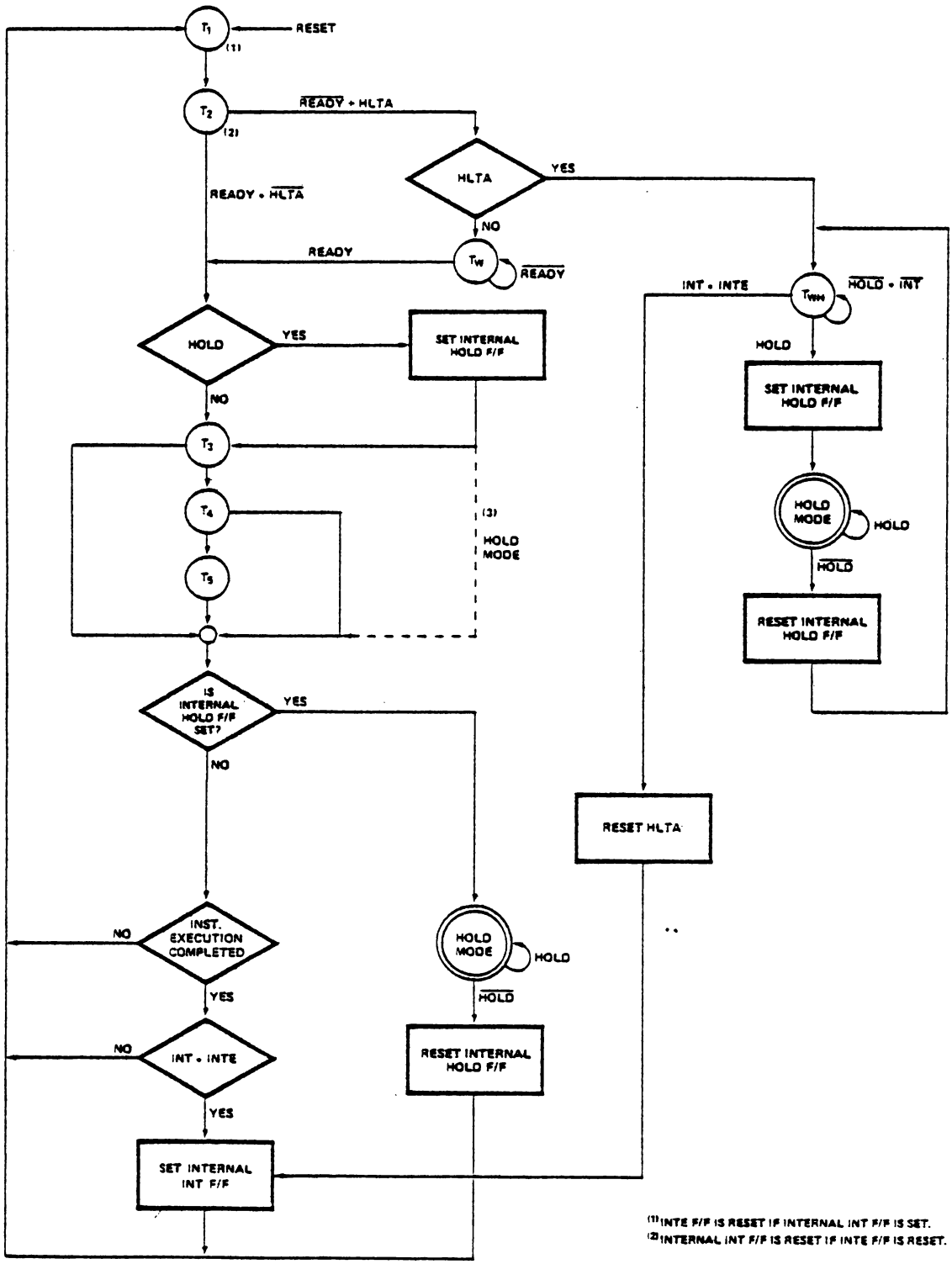



Figure 2-4. CPU State Transition Diagram

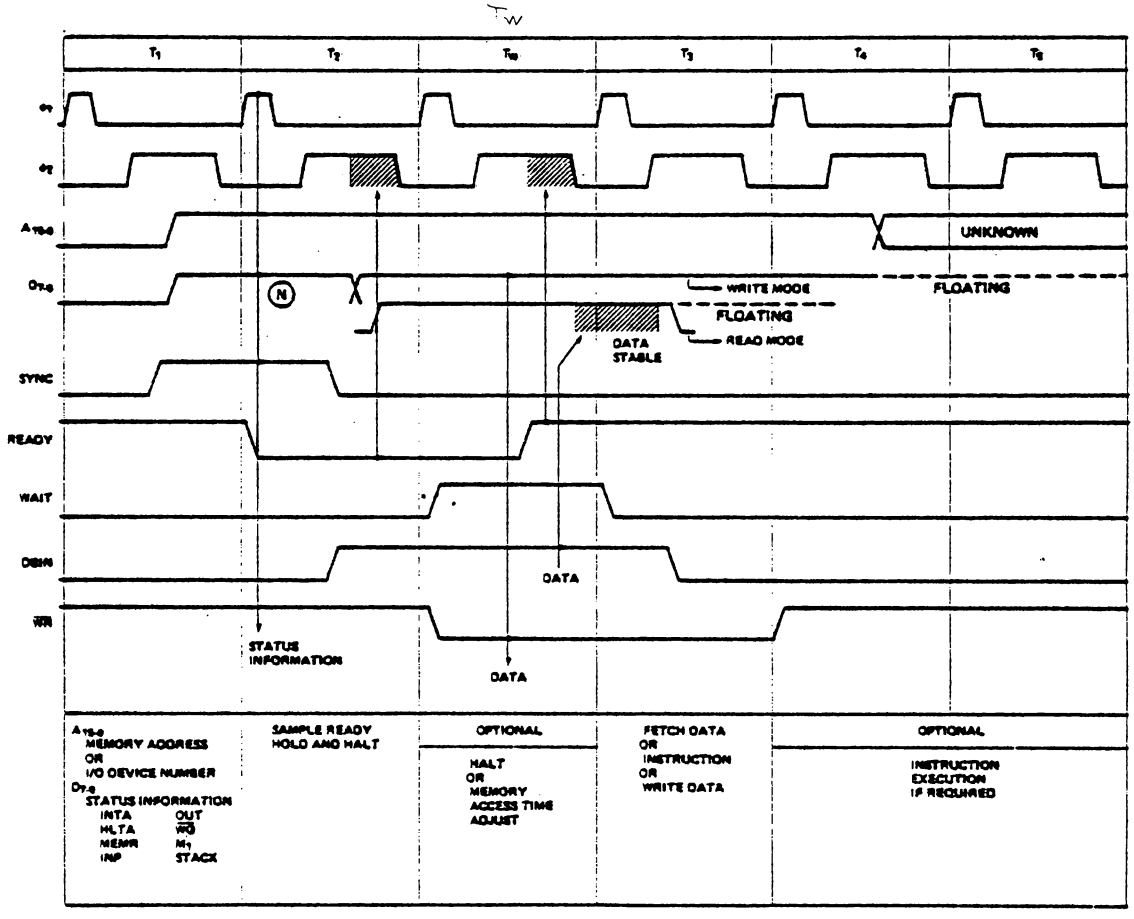
modular micros | martin research

The events that take place during the $T_3$ state are determined by the kind of machine cycle in progress. In a FETCH machine cycle, the processor interprets the data on its data bus as an instruction. During a MEMORY READ or a STACK READ, data on this bus is interpreted as a data word. The processor outputs data on this bus during a MEMORY WRITE machine cycle. During I/O operations, the processor may either transmit or receive data, depending on whether an OUTPUT or an INPUT operation is involved.

Figure 2-6 illustrates the timing that is characteristic of a data input operation. As shown, the low-to-high transition of $\phi_2$ during $T_2$ clears status information from the processor's data lines, preparing these lines for the receipt of incoming data. The data presented to the processor must have stabilized prior to both the "$\phi_1$—data set-up" interval ($t_{DS1}$), that precedes the falling edge of the $\phi_1$ pulse defining state $T_3$, and the "$\phi_2$—data set-up" interval ($t_{DS2}$), that precedes the rising edge of $\phi_2$ in state $T_3$. This same data must remain stable during the "data hold" interval ($t_{DH}$) that occurs following the rising edge of the $\phi_2$ pulse. Data placed on these lines by memory or by other external devices will be sampled during $T_3$.

During the input of data to the processor, the 8080 generates a DBIN signal which should be used externally to enable the transfer. Machine cycles in which DBIN is available include: FETCH, MEMORY READ, STACK READ, and INTERRUPT. DBIN is initiated by the rising edge of $\phi_2$ during state T2 and terminated by the corresponding edge of $\phi_2$ during $T_3$. Any $T_W$ phases intervening between $T_2$ and $T_3$ will therefore extend DBIN by one or more clock periods.

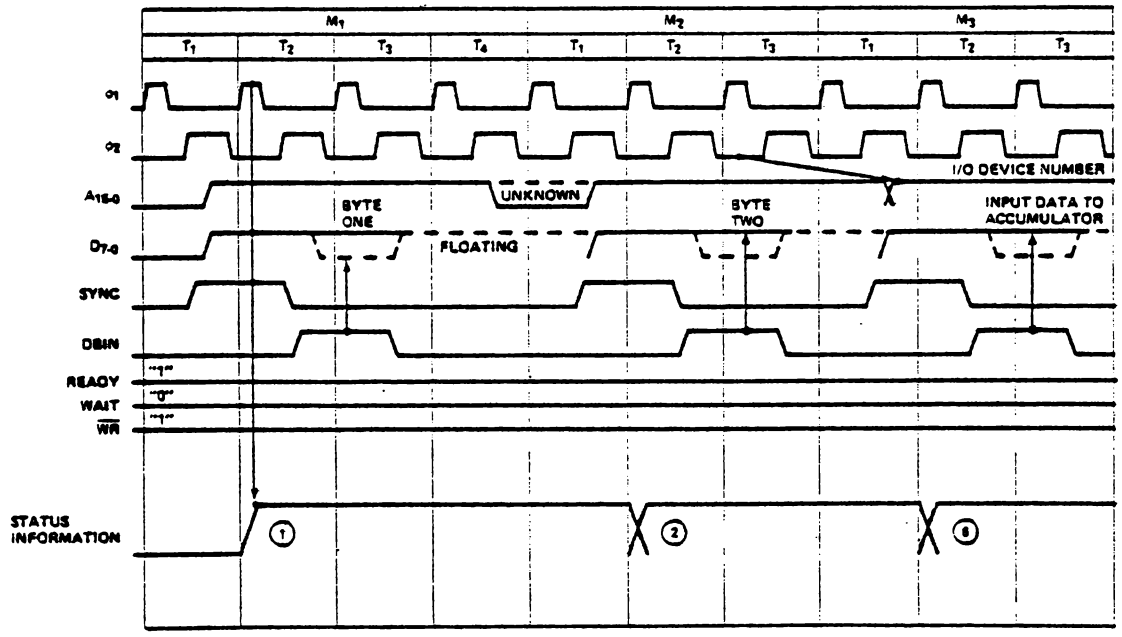
Figure 2-7 shows the timing of a machine cycle in which the processor outputs data. Output data may be destined either for memory or for peripherals. The rising edge of $\phi_2$ within state $T_2$ clears status information from the CPU's data lines, and loads in the data which is to be output to external devices. This substitution takes place within the




NOTE: (N) Refer to Status Word Chart

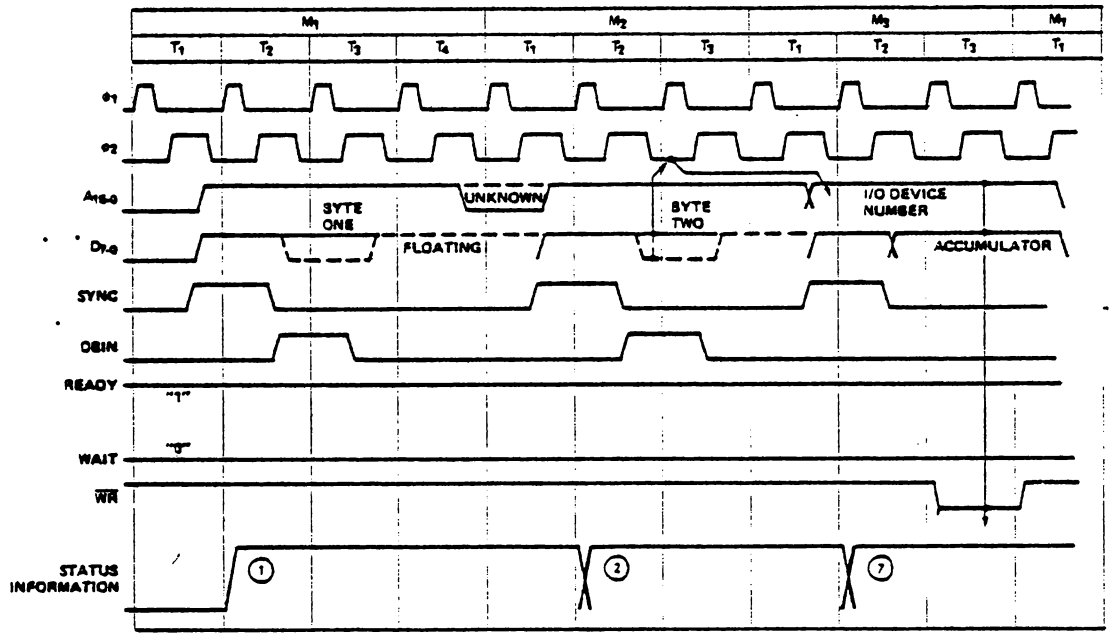
Figure 2-5. Basic 8080 Instruction Cycle

modular micros martin research




NOTE: (N) Refer to Status Word Chart

Figure 2-6. Input Instruction Cycle




NOTE: (N) Refer to Status Word Chart

Figure 2-7. Output Instruction Cycle

modular micros → martin research →

"data output delay" interval ($t_{DD}$) following the $\phi_2$ clock's leading edge. Data on the bus remains stable throughout the remainder of the machine cycle, until replaced by updated status information in the subsequent $T_1$ state. Observe that a READY signal is necessary for completion of an OUTPUT machine cycle. Unless such an indication is present, the processor enters the $T_W$ state, following the $T_2$ state. Data on the output lines remains stable in the interim, and the processing cycle will not proceed until the READY line again goes high.

The 8080 CPU generates a $\overline{WR}$ output for the synchronization of external transfers, during those machine cycles in which the processor outputs data. These include MEMORY WRITE, STACK WRITE, and OUTPUT. The negative-going leading edge of $\overline{WR}$ is referenced to the rising edge of the first $\phi_1$ clock pulse following $T_2$, and occurs within a brief delay ($t_{DC}$) of that event. $\overline{WR}$ remains low until re-triggered by the leading edge of $\phi_1$ during the state following $T_3$. Note that any $T_W$ states intervening between $T_2$ and $T_3$ of the output machine cycle will necessarily extend $\overline{WR}$, in much the same way that DBIN is affected during data input operations.

All processor machine cycles consist of at least three states: $T_1$, $T_2$, and $T_3$ as just described. If the processor has to wait for a response from the peripheral or memory with which it is communicating, then the machine cycle may also contain one or more $T_W$ states. During the three basic states, data is transferred to or from the processor.

After the $T_3$ state, however, it becomes difficult to generalize. $T_4$ and $T_5$ states are available, if the execution of a particular instruction requires them. But not all machine cycles make use of these states. It depends upon the kind of instruction being executed, and on the particular machine cycle within the instruction cycle. The processor will terminate any machine cycle as soon as its processing activities are completed, rather than proceeding through the $T_4$ and $T_5$ states every time. Thus the 8080 may exit a machine cycle following the $T_3$, the $T_4$, or the $T_5$ state and proceed directly to the $T_1$ state of the next machine cycle.

| STATE | ASSOCIATED ACTIVITIES |
|---|---|
| $T_1$ | A memory address or I/O device number is placed on the Address Bus ($A_{15-0}$); status information is placed on Data Bus ($D_{7-0}$). |
| $T_2$ | The CPU samples the READY and HOLD inputs and checks for halt instruction. |
| TW (optional) | Processor enters wait state if READY is low or if HALT instruction has been executed. |
| T3 | An instruction byte (FETCH machine cycle), data byte (MEMORY READ, STACK READ) or interrupt instruction (INTERRUPT machine cycle) is input to the CPU from the Data Bus; or a data byte (MEMORY WRITE, STACK WRITE or OUTPUT machine cycle) is output onto the data bus. |
| T4 T5 (optional) | States $T_4$ and $T_5$ are available if the execution of a particular instruction requires them; if not, the CPU may skip one or both of them. $T_4$ and $T_5$ are only used for internal processor operations. |

Table 2-2. State Definitions

modular micros → martin research →

## INTERRUPT SEQUENCES

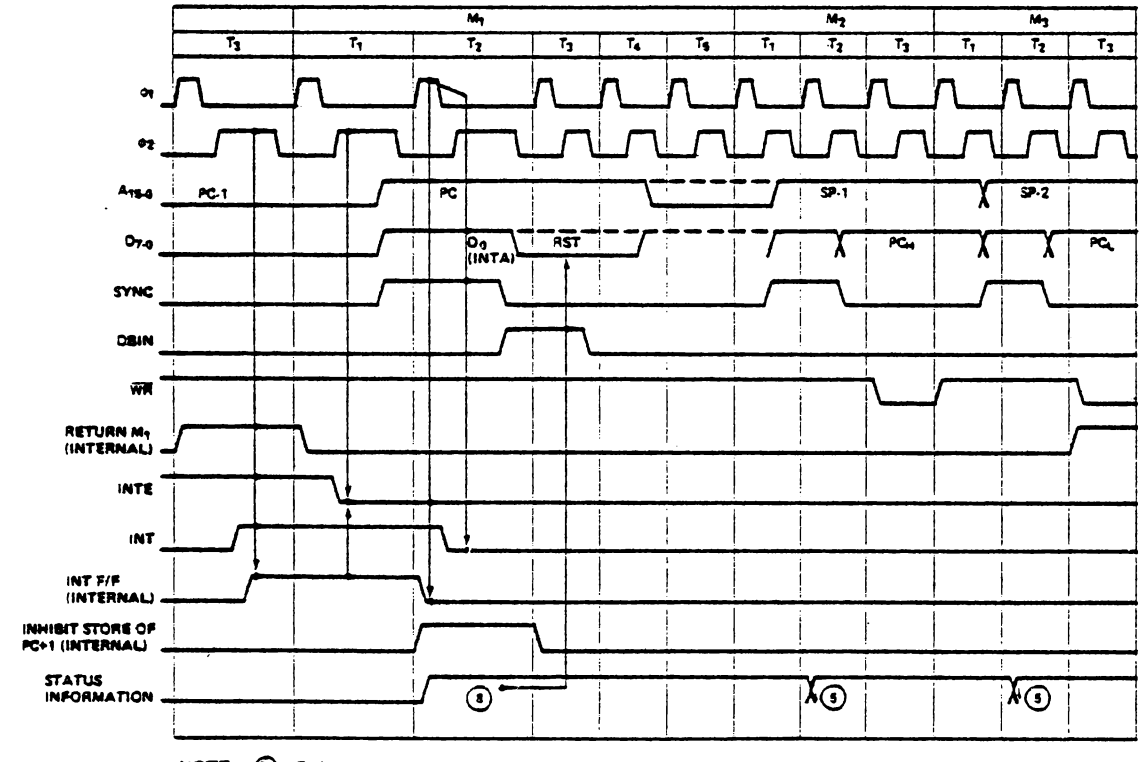
The 8080 has the built-in capacity to handle external interrupt requests. A peripheral device can initiate an interrupt simply by driving the processor's interrupt (INT) line high.

The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. As Figure 2-8 shows, an interrupt request (INT) arriving during the time that the interrupt enable line (INTE) is high, acts in coincidence with the $\phi_2$ clock to set the internal interrupt latch. This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is completed before the interrupt can be processed.

The INTERRUPT machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary FETCH machine cycle in most respects. The $M_1$ status bit is transmitted as usual during the SYNC interval. It is accompanied, however, by an INTA status bit ($D_0$) which acknowledges the external request. The contents of the program counter are latched onto the CPU's address lines during $T_1$, but the counter itself is not incremented during the INTERRUPT machine cycle, as it otherwise would be. In this way, the pre-interrupt status of the program counter is preserved, so that data in the counter may be restored by the interrupted program after the interrupt request has been processed.

The interrupt cycle is otherwise indistinguishable from an ordinary FETCH machine cycle. The processor itself takes no further special action. It is the responsibility of the peripheral logic to see that an eight-bit interrupt instruction is "jammed" onto the processor's data bus during state $T_3$. In a typical system, this means that the data-in bus from memory must be temporarily disconnected from the processor's main data bus, so that the interrupting device can command the main bus without interference.

The 8080's instruction set provides a special one-byte call which facilitates the processing of interrupts (the ordinary program Call takes three bytes). This is the RESTART instruction (RST). A variable three-bit field embedded in the eight-bit field of the RST enables the interrupting device to direct a Call to one of eight fixed memory locations. The decimal addresses of these dedicated locations are: 0, 8, 16, 24, 32, 40, 48, and 56. Any of these addresses may be used to store the first instruction(s) of a routine designed to service the requirements of an interrupting device. Since the (RST) is a call, completion of the instruction also stores the old program counter contents on the STACK.




NOTE: (N) Refer to Status Word Chart

Figure 2-8. Interrupt Timing

modular micros → martin research →

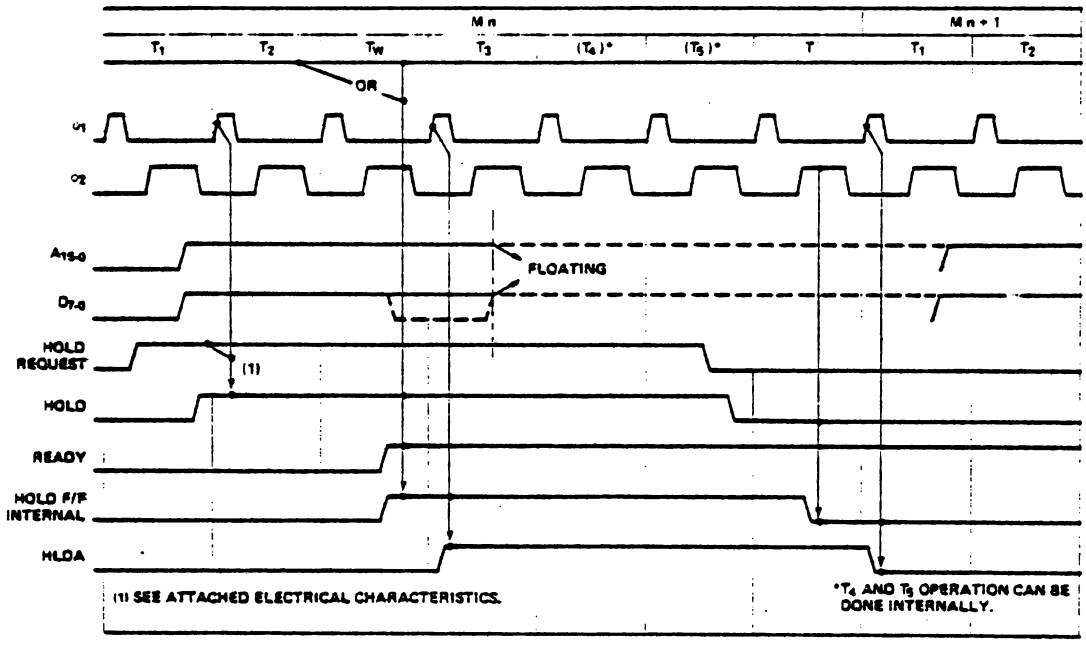



Figure 2-9. HOLD Operation (Read Mode)

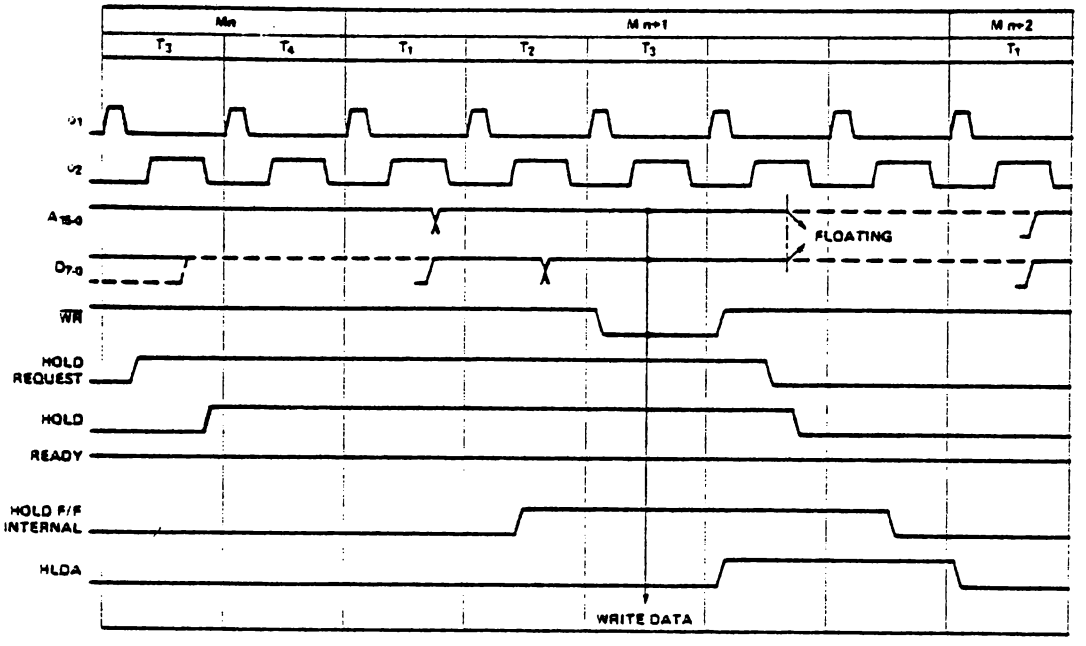



Figure 2-10. HOLD Operation (Write Mode)

modular micros → martin research

## HOLD SEQUENCES

The 8080A CPU contains provisions for Direct Memory Access (DMA) operations. By applying a HOLD to the appropriate control pin on the processor, an external device can cause the CPU to suspend its normal operations and relinquish control of the address and data busses. The processor responds to a request of this kind by floating its address to other devices sharing the busses. At the same time, the processor acknowledges the HOLD by placing a high on its HLDA outpin pin. During an acknowledged HOLD, the address and data busses are under control of the peripheral which originated the request, enabling it to conduct memory transfers without processor intervention.

Like the interrupt, the HOLD input is synchronized internally. A HOLD signal must be stable prior to the "Hold set-up" interval ($t_{HS}$), that precedes the rising edge of $\phi_2$.

Figures 2-9 and 2-10 illustrate the timing involved in HOLD operations. Note the delay between the asynchronous HOLD REQUEST and the re-clocked HOLD. As shown in the diagram, a coincidence of the READY, the HOLD, and the $\phi_2$ clocks sets the internal hold latch. Setting the latch enables the subsequent rising edge of the $\phi_1$ clock pulse to trigger the HLDA output.

Acknowledgement of the HOLD REQUEST precedes slightly the actual floating of the processor's address and data lines. The processor acknowledges a HOLD at the beginning of $T_3$, if a read or an input machine cycle is in progress (see Figure 2-9). Otherwise, acknowledgement is deferred until the beginning of the state following $T_3$ (see Figure 2-10). In both cases, however, the HLDA goes high within a specified delay ($t_{DC}$) of the rising edge of the selected $\phi_1$ clock pulse. Address and data lines are floated within a brief delay after the rising edge of the next $\phi_2$ clock pulse. This relationship is also shown in the diagrams.

To all outward appearances, the processor has suspended its operations once the address and data busses are floated. Internally, however, certain functions may continue. If a HOLD REQUEST is acknowledged at $T_3$, and if the processor is in the middle of a machine cycle which requires four or more states to complete, the CPU proceeds through $T_4$ and $T_5$ before coming to a rest. Not until the end of the machine cycle is reached will processing activities cease. Internal processing is thus permitted to overlap the external DMA transfer, improving both the efficiency and the speed of the entire system.

The processor exits the holding state through a sequence similar to that by which it entered. A HOLD REQUEST is terminated asynchronously when the external device has completed its data transfer. The HLDA output returns to a low level following the leading edge of the next $\phi 1$ clock pulse. Normal processing resumes with the machine cycle following the last cycle that was executed.

## HALT SEQUENCES

When a halt instruction (HLT) is executed, the CPU enters the halt state ($T_{WH}$) after state $T_2$ of the next machine cycle, as shown in Figure 2-11. There are only three ways in which the 8080 can exit the halt state:

- A high on the RESET line will always reset the 8080 to state $T_1$; RESET also clears the program counter.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next $\phi_1$ clock pulse.
- An interrupt (i.e., INT goes high while INTE is enabled) will cause the 8080 to exit the Halt state and enter state $T_1$ on the rising edge of the next $\phi_1$ clock pulse. NOTE: The interrupt enable (INTE) flag must be set when the halt state is entered; otherwise, the 8080 will only be able to exit via a RESET signal.

Figure 2-12 illustrates halt sequencing in flow chart form.

## START-UP OF THE 8080 CPU

When power is applied initially to the 8080, the processor begins operating immediately. The contents of its program counter, stack pointer, and the other working registers are naturally subject to random factors and cannot be specified. For this reason, it will be necessary to begin the power-up sequence with RESET.

An external RESET signal of three clock period duration (minimum) restores the processor's internal program counter to zero. Program execution thus begins with memory location zero, following a RESET. Systems which require the processor to wait for an explicit start-up signal will store a halt instruction (EI, HLT) in the first two locations. A manual or an automatic INTERRUPT will be used for starting. In other systems, the processor may begin executing its stored program immediately. Note, however, that the RESET has no effect on status flags, or on any of the processor's working registers (accumulator, registers, or stack pointer). The contents of these registers remain indeterminate, until initialized explicitly by the program.

modular micros martin research

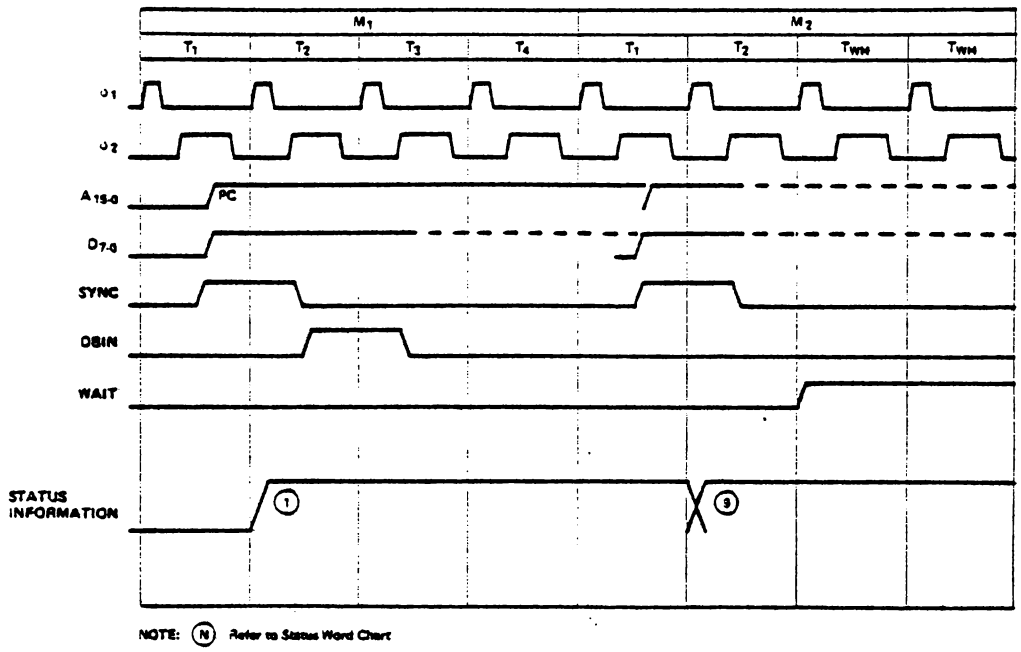



Figure 2-11. HALT Timing

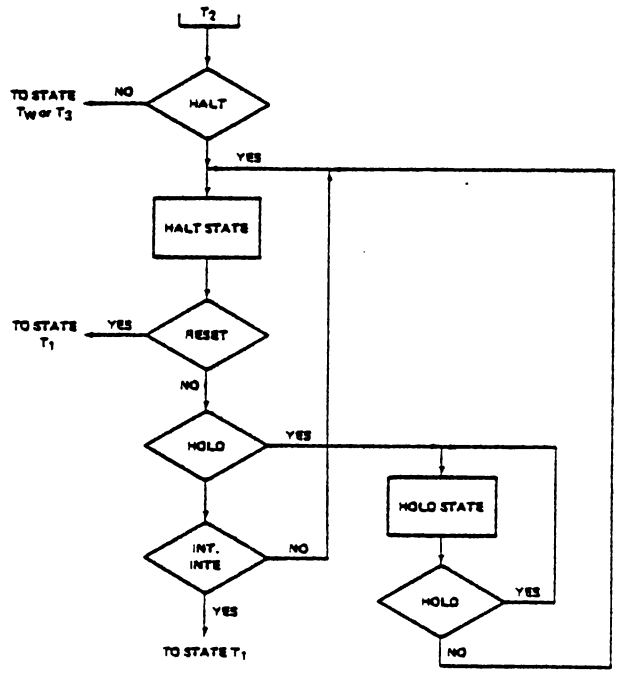



Figure 2-12. HALT Sequence Flow Chart.

modular micros
martin research

NOTES:

1. The first memory cycle (M1) is always an instruction fetch; the first (or only) byte, containing the op code, is fetched during this cycle.

2. If the READY input from memory is not high during T2 of each memory cycle, the processor will enter a wait state (TW) until READY is sampled as high.

3. States T4 and T5 are present, as required, for operations which are completely internal to the CPU. The contents of the internal bus during T4 and T5 are available at the data bus; this is designed for testing purposes only. An "X" denotes that the state is present, but is only used for such internal operations as instruction decoding.

4. Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

5. These states are skipped.

6. Memory read sub-cycles; an instruction or data word will be read.

7. Memory write sub-cycle.

8. The READY signal is not required during the second and third sub-cycles (M2 and M3). The HOLD signal is accepted during M2 and M3. The SYNC signal is not generated during M2 and M3. During the execution of DAD, M2 and M3 are required for an internal register-pair add; memory is not referenced.

9. The results of these arithmetic, logical or rotate instructions are not moved into the accumulator (A) until state T2 of the next instruction cycle. That is, A is loaded while the next instruction is being fetched; this overlapping of operations allows for faster processing.

10. If the value of the least significant 4-bits of the accumulator is greater than 9 or if the auxiliary carry bit is set, 6 is added to the accumulator. If the value of the most significant 4-bits of the accumulator is now greater than 9, or if the carry bit is set, 6 is added to the most significant 4-bits of the accumulator.

11. This represents the first sub-cycle (the instruction fetch) of the next instruction cycle.

12. If the condition was met, the contents of the register pair WZ are output on the address lines ($A_{0-15}$) instead of the contents of the program counter (PC).

13. If the condition was not met, sub-cycles M4 and M5 are skipped; the processor instead proceeds immediately to the instruction fetch (M1) of the next instruction cycle.

14. If the condition was not met, sub-cycles M2 and M3 are skipped; the processor instead proceeds immediately to the instruction fetch (M1) of the next instruction cycle.

15. Stack read sub-cycle.

16. Stack write sub-cycle.

17.

| CONDITION | CCC |
|---|---|
| NZ – not zero (Z = 0) | 000 |
| Z – zero (Z = 1) | 001 |
| NC – no carry (CY = 0) | 010 |
| C – carry (CY = 1) | 011 |
| PO – parity odd (P = 0) | 100 |
| PE – parity even (P = 1) | 101 |
| P – plus (S = 0) | 110 |
| M – minus (S = 1) | 111 |

18. I/O sub-cycle: the I/O port's 8-bit select code is duplicated on address lines 0-7 ($A_{0-7}$) and 8-15 ($A_{8-15}$).

19. Output sub-cycle.

20. The processor will remain idle in the halt state until an interrupt, a reset or a hold is accepted. When a hold request is accepted, the CPU enters the hold mode; after the hold mode is terminated, the processor returns to the halt state. After a reset is accepted, the processor begins execution at memory location zero. After an interrupt is accepted, the processor executes the instruction forced onto the data bus (usually a restart instruction).

| SSS or DDD | Value | rp | Value |
|---|---|---|---|
| A | 111 | B | 00 |
| B | 000 | D | 01 |
| C | 001 | H | 10 |
| D | 010 | SP | 11 |
| E | 011 | | |
| H | 100 | | |
| L | 101 | | |

modular micros | martin research

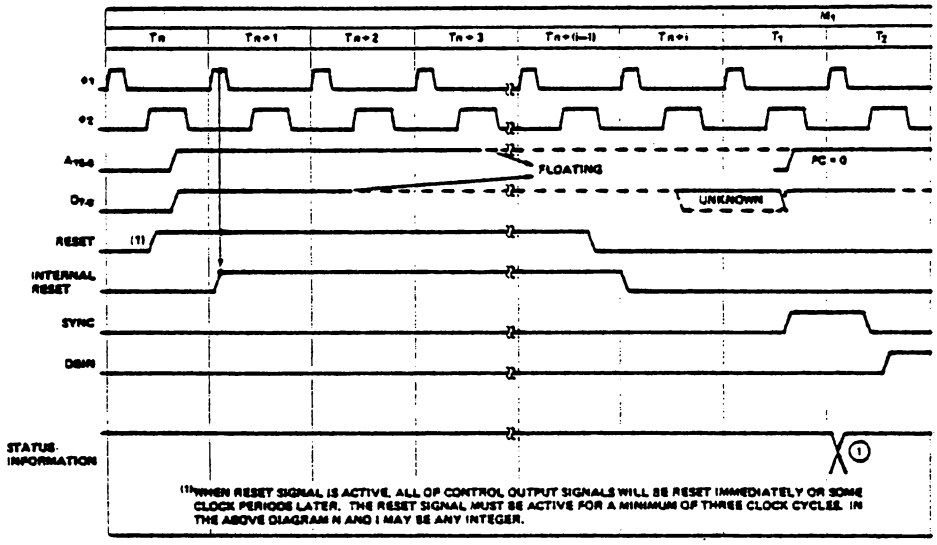



Figure 2-13. Reset.

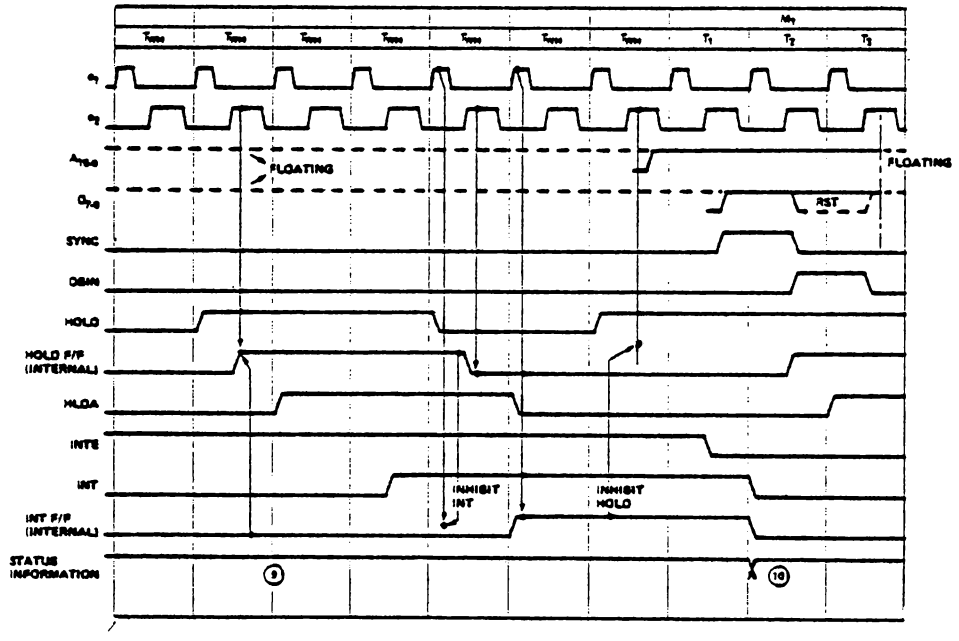



Figure 2-14. Relation between HOLD and INT in the HALT State.

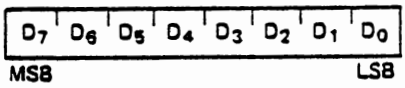
modular micros martin research

## THE 8080 INSTRUCTION SET

The 8080 can directly address up to 65,536 bytes of memory, which may consist of both read-only memory (ROM) elements and random-access memory (RAM) elements (read/write memory).

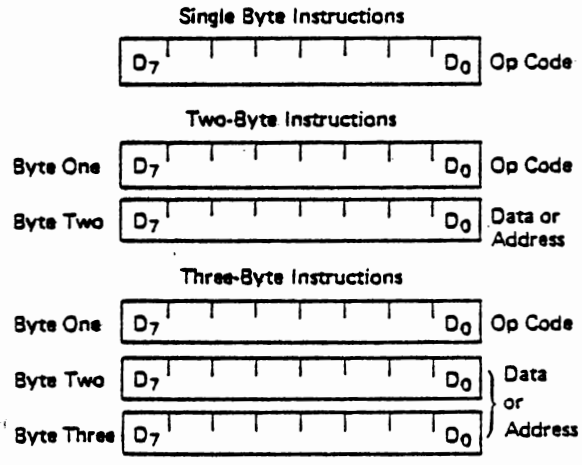
Data in the 8080 is stored in the form of 8-bit binary integers:

DATA WORD




When a register or data word contains a binary number, it is necessary to establish the order in which the bits of the number are written. In the Intel 8080, BIT 0 is referred to as the Least Significant Bit (LSB), and BIT 7 (of an 8 bit number) is referred to as the Most Significant Bit (MSB).

The 8080 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instructions. The exact instruction format will depend on the particular operation to be executed.




### Addressing Modes:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers:

- Direct – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in byte 3).
- Register – The instruction specifies the register or register-pair in which the data is located.
- Register Indirect – The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- Immediate – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- Direct – The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- Register indirect – The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

### Condition Flags:

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition), or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise it is reset.

modular micros martin research

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions, subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

## Symbols and Abbreviations:

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

| SYMBOLS | MEANING |
|---|---|
| accumulator | Register A |
| addr | 16-bit address quantity |
| data | 8-bit data quantity |
| data 16 | 16-bit data quantity |
| byte 2 | The second byte of the instruction |
| byte 3 | The third byte of the instruction |
| port | 8-bit address of an I/O device |
| r,r1,r2 | One of the registers A,B,C,D,E,H,L |
| DDD,SSS | The bit pattern designating one of the registers A,B,C,D,E,H,L (DDD=destination, SSS=source): |

| DDD or SSS | REGISTER NAME |
|---|---|
| 111 | A |
| 000 | B |
| 001 | C |
| 010 | D |
| 011 | E |
| 100 | H |
| 101 | L |

| | |
|---|---|
| rp | One of the register pairs:<br>B represents the B,C pair with B as the high-order register and C as the low-order register;<br>D represents the D,E pair with D as the high-order register and E as the low-order register;<br>H represents the H,L pair with H as the high-order register and L as the low-order register;<br>SP represents the 16-bit stack pointer register. |
| RP | The bit pattern designating one of the register pairs B,D,H,SP: |

| RP | REGISTER PAIR |
|---|---|
| 00 | B-C |
| 01 | D-E |
| 10 | H-L |
| 11 | SP |

| | |
|---|---|
| rh | The first (high-order) register of a designated register pair. |
| rl | The second (low-order) register of a designated register pair. |
| PC | 16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively). |
| SP | 16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively). |
| $r_m$ | Bit m of the register r (bits are number 7 through 0 from left to right). |
| Z,S,P,CY,AC | The condition flags:<br>Zero,<br>Sign,<br>Parity,<br>Carry,<br>and Auxiliary Carry, respectively. |
| ( ) | The contents of the memory location or registers enclosed in the parentheses. |
| ← | "Is transferred to" |
| ∧ | Logical AND |
| ∀ | Exclusive OR |
| ∨ | Inclusive OR |
| + | Addition |
| − | Two's complement subtraction |
| * | Multiplication |
| ↔ | "Is exchanged with" |
| — | The one's complement (e.g., $(\overline{A})$) |
| n | The restart number 0 through 7 |
| NNN | The binary representation 000 through 111 for restart number 0 through 7 respectively. |

The following pages provide a detailed description of the instruction set of the 8080.

modular micros martin research

modular micros

martin research

ADD ou SS° register name

111 A
000 B
001 C
010 D
011 E
100 H
101 L

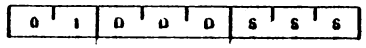
## Data Transfer Group:

This group of instructions transfers data to and from registers and memory. Condition flags are not affected by any instruction in this group.

**MOV r1, r2** (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

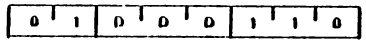



Cycles: 1
States: 5
Addressing: register
Flags: none

**MOV r, M** (Move from memory)

(r) ← ((H)(L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

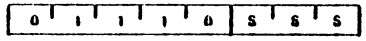



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

**MOV M, r** (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

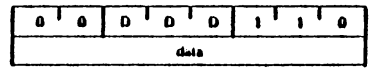



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

**MVI r, data** (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.




Cycles: 2
States: 7
Addressing: immediate
Flags: none

**MVI M, data** (Move to memory immediate)

((H) (L)) ← (byte 2)

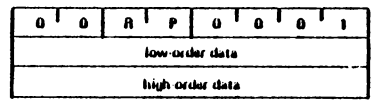
The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

Cycles: 3
States: 10
Addressing: immed./reg. indirect
Flags: none

**LXI rp, data 16** (Load register pair immediate)

(rh) ← (byte 3),
(rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

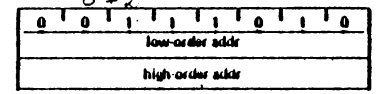



Cycles: 3
States: 10
Addressing: immediate
Flags: none

**LDA addr** (Load Accumulator direct)

(A) ← ((byte 3)(byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

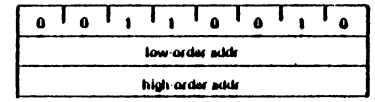



Cycles: 4
States: 13
Addressing: direct
Flags: none

**STA addr** (Store Accumulator direct)

((byte 3)(byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

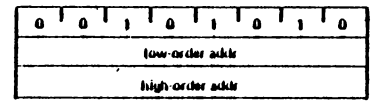



Cycles: 4
States: 13
Addressing: direct
Flags: none

**LHLD addr** (Load H and L direct)

(L) ← ((byte 3)(byte 2))
(H) ← ((byte 3)(byte 2) + 1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

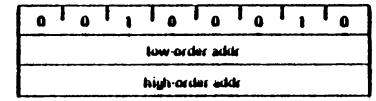



Cycles: 5
States: 16
Addressing: direct
Flags: none

**SHLD addr** (Store H and L direct)

((byte 3)(byte 2)) ← (L)
((byte 3)(byte 2) + 1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

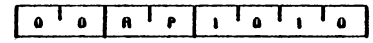



Cycles: 5
States: 16
Addressing: direct
Flags: none

**LDAX rp** (Load accumulator indirect)

(A) ← ((rp))

The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

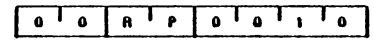



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

**STAX rp** (Store accumulator indirect)

((rp)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

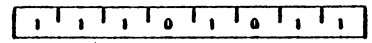



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

**XCHG** (Exchange H and L with D and E)

(H) ↔ (D)
(L) ↔ (E)

The contents of registers H and L are exchanged with the contents of registers D and E.




Cycles: 1
States: 4
Addressing: register
Flags: none

## Arithmetic Group:

This group of instructions performs arithmetic operations on data in registers and memory.

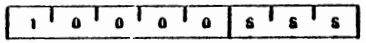
Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

**ADD r** (Add Register)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result is placed in the accumulator.

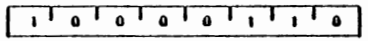



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**ADD M** (Add memory)

(A) ← (A) + ((H) (L))

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

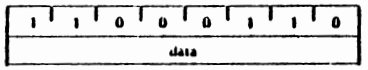



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**ADI data** (Add immediate)

(A) ← (A) + (byte 2)

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.

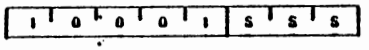



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**ADC r** (Add Register with carry)

(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

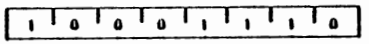



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**ADC M** (Add memory with carry)

(A) ← (A) + ((H) (L)) + (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

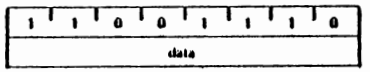



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**ACI data** (Add immediate with carry)

(A) ← (A) + (byte 2) + (CY)

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

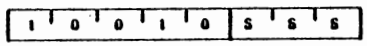



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**SUB r** (Subtract Register)

(A) ← (A) − (r)

The content of register r is subtracted from the contents of the accumulator. The result is placed in the accumulator.

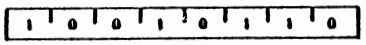



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**SUB M** (Subtract memory)

(A) ← (A) − ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

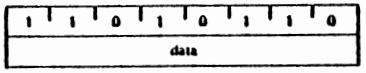



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**SUI data** (Subtract immediate)

(A) ← (A) − (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

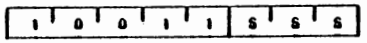



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**SBB r** (Subtract Register with borrow)

(A) ← (A) − (r) − (CY)

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

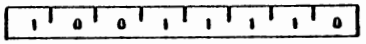



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**SBB M** (Subtract memory with borrow)

(A) ← (A) − ((H) (L)) − (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

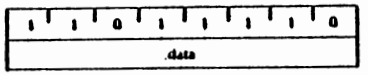



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**SBI data** (Subtract immediate with borrow)

(A) ← (A) − (byte 2) − (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

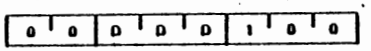



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**INR r** (Increment Register)

(r) ← (r) + 1

The content of register r is incremented by one. Note: All condition flags except CY are affected.

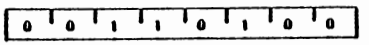



Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

**INR M** (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags except CY are affected.

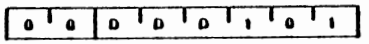



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

**DCR r** (Decrement Register)

(r) ← (r) − 1

The content of register r is decremented by one. Note: All condition flags except CY are affected.



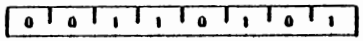

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

modular micros ⇄ martin research

**DCR M** (Decrement memory)

((H) (L)) ← ((H) (L)) − 1

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags except CY are affected.

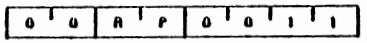



Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

**INX rp** (Increment register pair)

(rh) (rl) ← (rh) (rl) + 1

The content of the register pair rp is incremented by one. Note: No condition flags are affected.

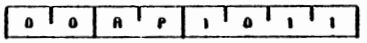



Cycles: 1
States: 6
Addressing: register
Flags: none

**DCX rp** (Decrement register pair)

(rh) (rl) ← (rh) (rl) − 1

The content of the register pair rp is decremented by one. Note: No condition flags are affected.

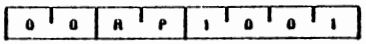



Cycles: 1
States: 6
Addressing: register
Flags: none

**DAD rp** (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (rl)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: Only the CY flag is affected. It is set if there is a carry out of the double precision add; otherwise it is reset.



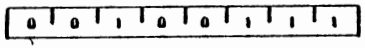

Cycles: 3
States: 10
Addressing: register
Flags: CY

**DAA** (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.




Cycles: 1
States: 4
Flags: Z,S,P,CY,AC

## Logical Group:

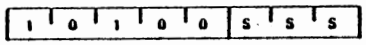
This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

**ANA r** (AND Register)

(A) ← (A) ∧ (r)

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. The CY flag is cleared.

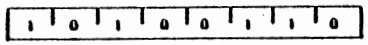



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**ANA M** (AND memory)

(A) ← (A) ∧ ((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. The CY flag is cleared.

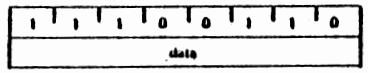



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**ANI data** (AND immediate)

(A) ← (A) ∧ (byte 2)

The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

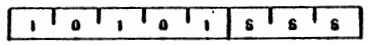



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**XRA r** (Exclusive OR Register)

(A) ← (A) ∀ (r)

The content of register r is exclusive-or'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

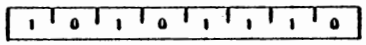



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**XRA M** (Exclusive OR Memory)

(A) ← (A) ∀ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

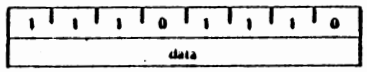



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**XRI data** (Exclusive OR immediate)

(A) ← (A) ∀ (byte 2)

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

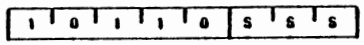



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**ORA r** (OR Register)

(A) ← (A) ∨ (r)

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

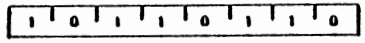



Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

**ORA M** (OR memory)

(A) ← (A) ∨ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

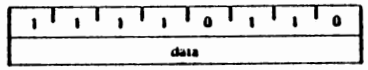



Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**ORI data** (OR immediate)

(A) ← (A) ∨ (byte 2)

The content of the second byte of the instruction is inclusive OR'd with the content of the accumulator. The result is placed in the accumulator. The CY and AC flags are cleared.

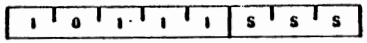



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**CMP r** (Compare Register)

(A) − (r)

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).



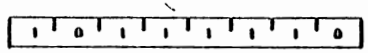

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

modular micros
martin research

**CMP M** (Compare memory)

(A) − ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).

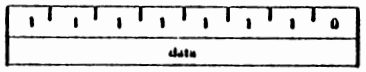


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

**CPI data** (Compare immediate)

(A) − (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).

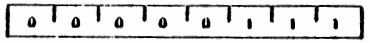



Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

**RLC** (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$ ; $(A_0) \leftarrow (A_7)$
$(CY) \leftarrow (A_7)$

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. Only the CY flag is affected.

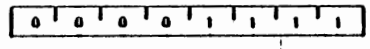


Cycles: 1
States: 4
Flags: CY

**RRC** (Rotate right)

$(A_n) \leftarrow (A_{n-1})$ ; $(A_7) \leftarrow (A_0)$
$(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. Only the CY flag is affected.

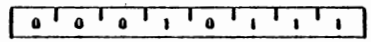


Cycles: 1
States: 4
Flags: CY

**RAL** (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$ ; $(CY) \leftarrow (A_7)$
$(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. Only the CY flag is affected.

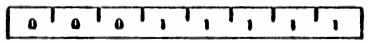


Cycles: 1
States: 4
Flags: CY

**RAR** (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$ ; $(CY) \leftarrow (A_0)$
$(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. Only the CY flag is affected.

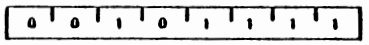


Cycles: 1
States: 4
Flags: CY

**CMA** (Complement accumulator)

$(A) \leftarrow \overline{(A)}$

The contents of the accumulator are complemented (zero bits become 1, one bits become 0). No flags are affected.

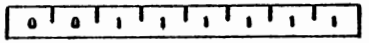


Cycles: 1
States: 4
Flags: none

**CMC** (Complement carry)

$(CY) \leftarrow \overline{(CY)}$

The CY flag is complemented. No other flags are affected.

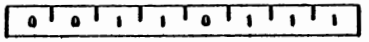


Cycles: 1
States: 4
Flags: CY

**STC** (Set carry)

(CY) ← 1

The CY flag is set to 1. No other flags are affected.



Cycles: 1
States: 4
Flags: CY

## Branch Group:

This group of instructions alter normal sequential program flow.

Condition flags are not affected by any instruction in this group.

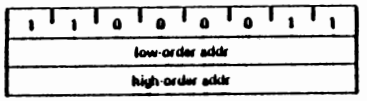
The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

| CONDITION | CCC |
|---|---|
| NZ – not zero (Z = 0) | 000 |
| Z – zero (Z = 1) | 001 |
| NC – no carry (CY = 0) | 010 |
| C – carry (CY = 1) | 011 |
| PO – parity odd (P = 0) | 100 |
| PE – parity even (P = 1) | 101 |
| P – plus (S = 0) | 110 |
| M – minus (S = 1) | 111 |

**JMP addr** (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



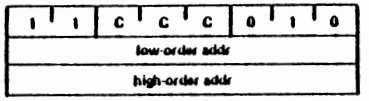

Cycles: 3
States: 10
Addressing: immediate
Flags: none

**Jcondition addr** (Conditional jump)

If (CCC),

(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

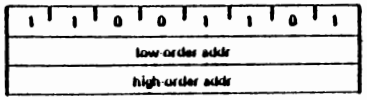



Cycles: 3
States: 10
Addressing: immediate
Flags: none

**CALL addr** (Call)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.




Cycles: 5
States: 17
Addressing: immediate/reg. indirect
Flags: none

modular micros ⇄ martin research

**Ccondition addr** (Condition call)

If (CCC),
((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

| 1 | 1 | C | C | C | 1 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| low-order addr | | | | | | | |
| high-order addr | | | | | | | |

Cycles: 3/6
States: 11/17
Addressing: immediate/reg. indirect
Flags: none

**RET** (Return)

(PCL) ← ((SP));
(PCH) ← ((SP) + 1);
(SP) ← (SP) + 2;

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2

| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

**Rcondition** (Conditional return)

If (CCC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

| 1 | 1 | C | C | C | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

**RST n** (Restart)

((SP) − 1) ← (PCH)
((SP) − 2) ← (PCL)
(SP) ← (SP) − 2
(PC) ← 8 • (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

| 1 | 1 | N | N | N | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | N | N | N | 0 | 0 | 0 |

Program Counter After Restart

**PCHL** (Jump H and L indirect – move H and L to PC)

(PCH) ← (H)
(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cycles: 1
States: 5
Addressing: register
Flags: none

## Stack, I/O, and Machine Control Group:

This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, condition flags are not affected by any instructions in this group.

**PUSH rp** (Push)

((SP) − 1) ← (rh)
((SP) − 2) ← (rl)
(SP) ← (SP) − 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Note: Register pair rp = SP may not be specified.

| 1 | 1 | R | P | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

**PUSH PSW** (Push processor status word)

((SP) − 1) ← (A)
((SP) − 2)$_0$ ← (CY), ((SP) − 2)$_1$ ← 1
((SP) − 2)$_2$ ← (P), ((SP) − 2)$_3$ ← 0
((SP) − 2)$_4$ ← (AC), ((SP) − 2)$_5$ ← 0
((SP) − 2)$_6$ ← (Z), ((SP) − 2)$_7$ ← (S)
(SP) ← (SP) − 2

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.

| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

**FLAG WORD**

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| S | Z | 0 | AC | 0 | P | 1 | CY |

**POP rp** (Pop)

(rl) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. Note: Register pair rp = SP may not be specified.

| 1 | 1 | R | P | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

**POP PSW** (Pop processor status word)

(CY) ← ((SP))$_0$
(P) ← ((SP))$_2$
(AC) ← ((SP))$_4$
(Z) ← ((SP))$_6$
(S) ← ((SP))$_7$
(A) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|

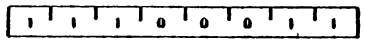
Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

modular micros
martin research

**XTHL** (Exchange stack top with H and L)

(L) ⟷ ((SP))
(H) ⟷ ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

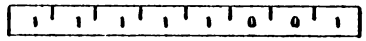


Cycles: 5
States: 18
Addressing: reg. indirect
Flags: none

**SPHL** (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

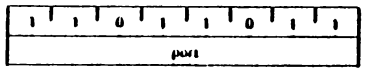


Cycles: 1
States: 5
Addressing: register
Flags: none

**IN port** (Input)

(A) ← (data)

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

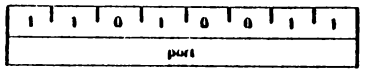



Cycles: 3
States: 10
Addressing: direct
Flags: none

**OUT port** (Output)

(data) ← (A)

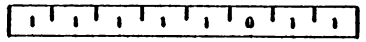
The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.




Cycles: 3
States: 10
Addressing: direct
Flags: none

**EI** (Enable interrupts)

The interrupt system is enabled following the execution of the next instruction.



Cycles: 1
States: 4
Flags: none

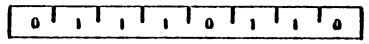
**DI** (Disable interrupts)

The interrupt system is disabled immediately following the execution of the DI instruction.

Cycles: 1
States: 4
Flags: none

**HLT** (Halt)

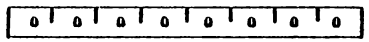
The processor is stopped. The registers and flags are unaffected.



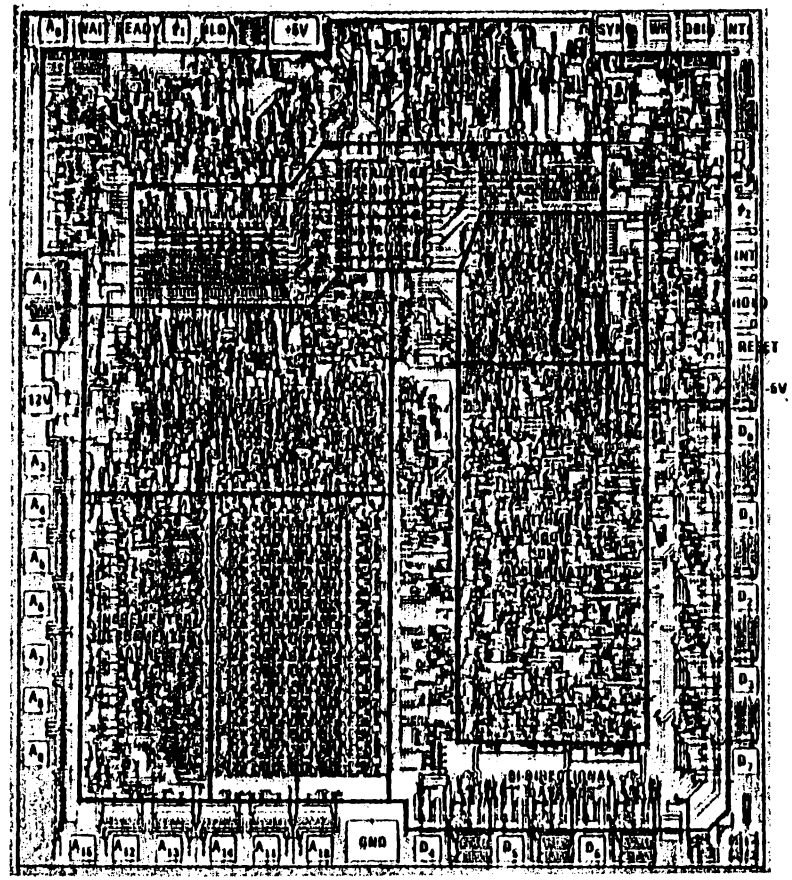
Cycles: 1
States: 7
Flags: none

**NOP** (No op)

No operation is performed. The registers and flags are unaffected.



Cycles: 1
States: 4
Flags: none




**8080 Photomicrograph With Pin Designations**

modular micros
martin research

## 8080A FUNCTIONAL PIN DEFINITION

The following describes the function of all of the 8080A I/O pins. Several of the descriptions refer to internal timing periods.

**$A_{15}$-$A_0$ (output three-state)**
ADDRESS BUS; the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. $A_0$ is the least significant address bit.

**$D_7$-$D_0$ (input/output three-state)**
DATA BUS; the data bus provides bi-directional communication between the CPU, memory, and I/O devices for instructions and data transfers. Also, during the first clock cycle of each machine cycle, the 8080A outputs a status word on the data bus that describes the current machine cycle. $D_0$ is the least significant bit.

**SYNC (output)**
SYNCHRONIZING SIGNAL; the SYNC pin provides a signal to indicate the beginning of each machine cycle.

**DBIN (output)**
DATA BUS IN; the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080A data bus from memory or I/O.

**READY (input)**
READY; the READY signal indicates to the 8080A that valid memory or input data is available on the 8080A data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080A does not receive a READY input, the 8080A will enter a WAIT state for as long as the READY line is low. READY can also be used to single step the CPU.

**WAIT (output)**
WAIT; the WAIT signal acknowledges that the CPU is in a WAIT state.

**$\overline{WR}$ (output)**
WRITE; the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active low ($\overline{WR}$ = 0).

**HOLD (input)**
HOLD; the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080A address and data bus as soon as the 8080A has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

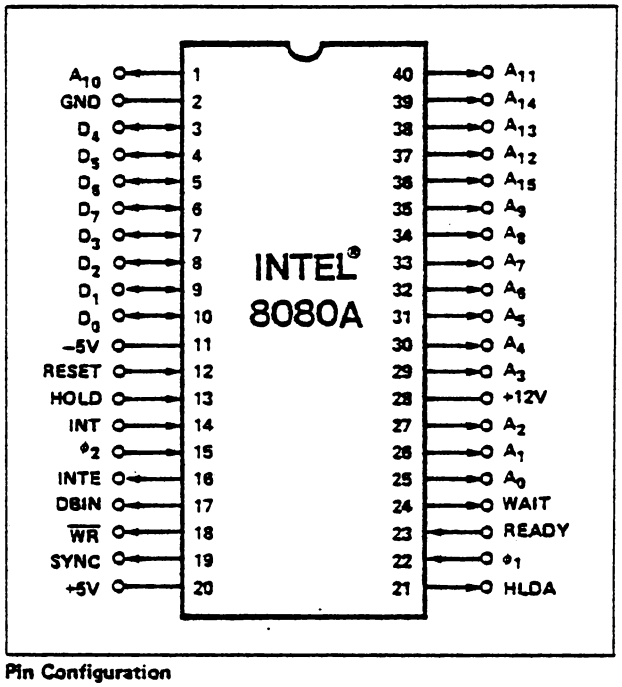
- the CPU is in the HALT state.
- the CPU is in the T2 or TW state and the READY signal is active.

As a result of entering the HOLD state the CPU ADDRESS BUS ($A_{15}$-$A_0$) and DATA BUS ($D_7$-$D_0$) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

**HLDA (output)**
HOLD ACKNOWLEDGE; the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at:

- T3 for READ memory or input.
- The Clock Period following T3 for WRITE memory or OUTPUT operation.

In either case, the HLDA signal appears after the rising edge of $\phi_1$ and high impedance occurs after the rising edge of $\phi_2$.



Pin Configuration

**INTE (output)**
INTERRUPT ENABLE; indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

**INT (input)**
INTERRUPT REQUEST; the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

**RESET (input)** [1]
RESET; while the RESET signal is activated, the content of the program counter is cleared. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, stack pointer, and registers are not cleared.

| | |
|---|---|
| $V_{SS}$ | Ground Reference. |
| $V_{DD}$ | +12 ± 5% Volts. |
| $V_{CC}$ | +5 ± 5% Volts. |
| $V_{BB}$ | -5 ±5% Volts (substrate bias). |
| $\phi_1$, $\phi_2$ | 2 externally supplied clock phases (non TTL compatible) |

modular micros martin research

## ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias . . . . . 0°C to +70° C
Storage Temperature . . . . . -65°C to +150°C
All Input or Output Voltages
With Respect to $V_{BB}$ . . . . . -0.3V to +20V
$V_{CC}$, $V_{DD}$ and $V_{SS}$ With Respect to $V_{BB}$ -0.3V to +20V
Power Dissipation . . . . . 1.5W

**COMMENT: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

## D.C. CHARACTERISTICS

$T_A$ = 0°C to 70°C, $V_{DD}$ = +12V ± 5%, $V_{CC}$ = +5V ± 5%, $V_{BB}$ = -5V ± 5%, $V_{SS}$ = 0V, Unless Otherwise Noted.

| Symbol | Parameter | Min. | Typ. | Max. | Unit | Test Condition |
|---|---|---|---|---|---|---|
| $V_{ILC}$ | Clock Input Low Voltage | $V_{SS}$-1 | | $V_{SS}$+0.8 | V | |
| $V_{IHC}$ | Clock Input High Voltage | 9.0 | | $V_{DD}$+1 | V | |
| $V_{IL}$ | Input Low Voltage | $V_{SS}$-1 | | $V_{SS}$+0.8 | V | |
| $V_{IH}$ | Input High Voltage | 3.3 | | $V_{CC}$+1 | V | |
| $V_{OL}$ | Output Low Voltage | | | 0.45 | V | $I_{OL}$ = 1.9mA on all outputs, |
| $V_{OH}$ | Output High Voltage | 3.7 | | | V | $I_{OH}$ = -150μA. |
| $I_{DD(AV)}$ | Avg. Power Supply Current ($V_{DD}$) | | 40 | 70 | mA | Operation |
| $I_{CC(AV)}$ | Avg. Power Supply Current ($V_{CC}$) | | 60 | 80 | mA | $T_{CY}$ = .48 μsec |
| $I_{BB(AV)}$ | Avg. Power Supply Current ($V_{BB}$) | | .01 | 1 | mA | |
| $I_{IL}$ | Input Leakage | | | ±10 | μA | $V_{SS} \leq V_{IN} \leq V_{CC}$ |
| $I_{CL}$ | Clock Leakage | | | ±10 | μA | $V_{SS} \leq V_{CLOCK} \leq V_{DD}$ |
| $I_{DL}$ [2] | Data Bus Leakage in Input Mode | | | -100<br>-2.0 | μA<br>mA | $V_{SS} \leq V_{IN} \leq V_{SS}$+0.8V<br>$V_{SS}$+0.8V $\leq V_{IN} \leq V_{CC}$ |
| $I_{FL}$ | Address and Data Bus Leakage During HOLD | | | +10<br>-100 | μA | $V_{ADDR/DATA} = V_{CC}$<br>$V_{ADDR/DATA} = V_{SS}$ + 0.45V |

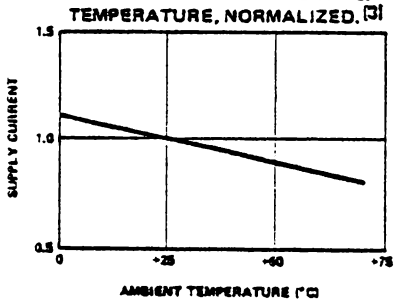
## CAPACITANCE

$T_A$ = 25°C $V_{CC} = V_{DD} = V_{SS}$ = 0V, $V_{BB}$ = -5V

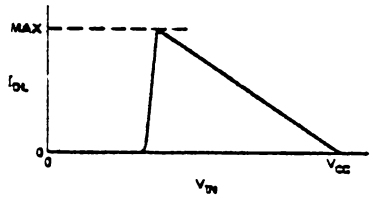
| Symbol | Parameter | Typ. | Max. | Unit | Test Condition |
|---|---|---|---|---|---|
| $C_\phi$ | Clock Capacitance | 17 | 25 | pf | $f_c$ = 1 MHz |
| $C_{IN}$ | Input Capacitance | 6 | 10 | pf | Unmeasured Pins |
| $C_{OUT}$ | Output Capacitance | 10 | 20 | pf | Returned to $V_{SS}$ |

NOTES:
1. The RESET signal must be active for a minimum of 3 clock cycles.
2. When DBIN is high and $V_{IN} > V_{IH}$ an internal active pull up will be switched onto the Data Bus.
3. $\Delta I$ supply / $\Delta T_A$ = -0.45%/°C.

TYPICAL SUPPLY CURRENT VS. TEMPERATURE, NORMALIZED. [3]




DATA BUS CHARACTERISTIC DURING DBIN




modular micros martin research

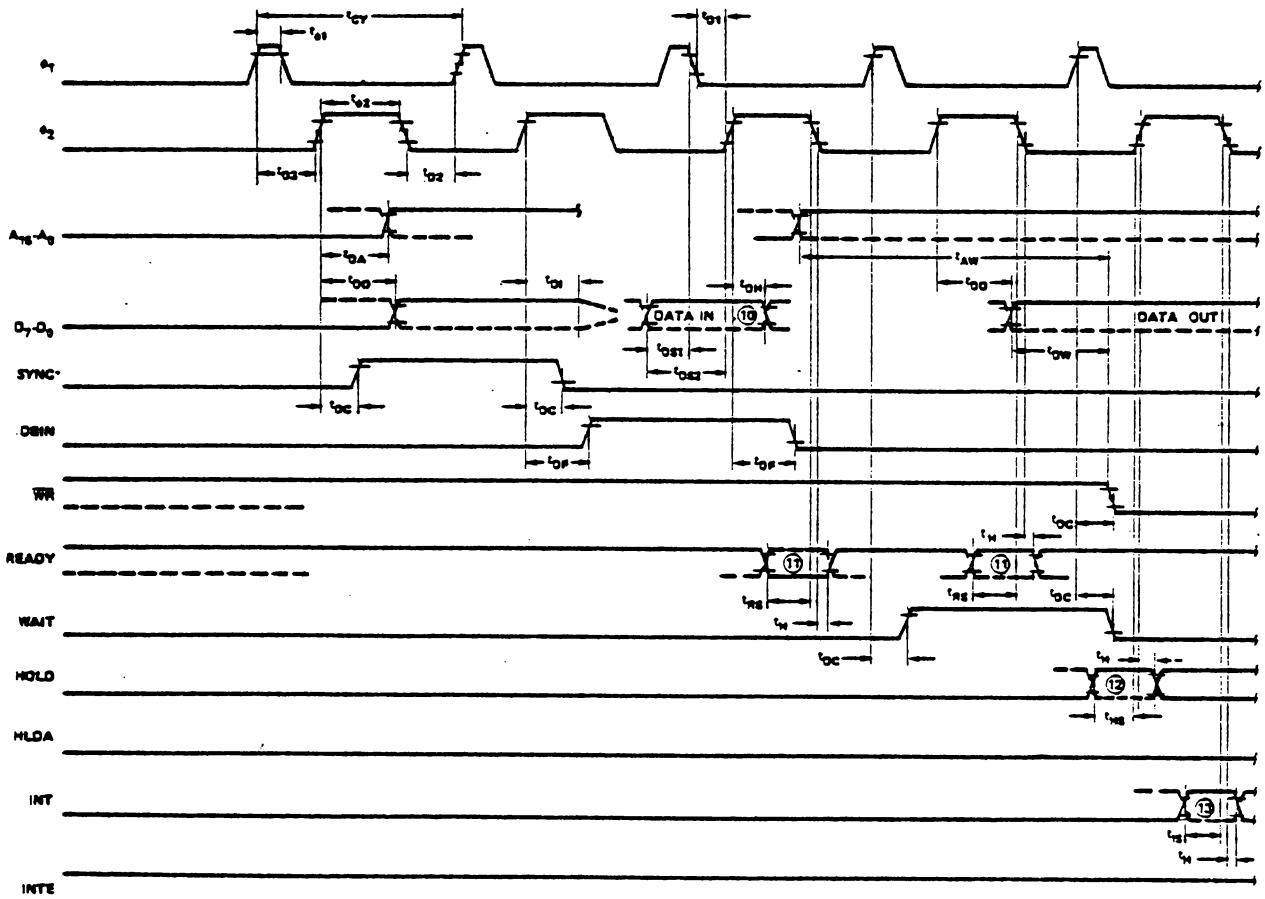
## A.C. CHARACTERISTICS

$T_A$ = 0°C to 70°C, $V_{DD}$ = +12V ± 5%, $V_{CC}$ = +5V ± 5%, $V_{BB}$ = -5V ± 5%, $V_{SS}$ = 0V, Unless Otherwise Noted

| Symbol | Parameter | Min. | Max. | Unit | Test Condition |
|---|---|---|---|---|---|
| $t_{CY}$[3] | Clock Period | 0.48 | 2.0 | μsec | |
| $t_r$, $t_f$ | Clock Rise and Fall Time | 0 | 50 | nsec | |
| $t_{\phi 1}$ | $\phi_1$ Pulse Width | 60 | | nsec | |
| $t_{\phi 2}$ | $\phi_2$ Pulse Width | 220 | | nsec | |
| $t_{D1}$ | Delay $\phi_1$ to $\phi_2$ | 0 | | nsec | |
| $t_{D2}$ | Delay $\phi_2$ to $\phi_1$ | 70 | | nsec | |
| $t_{D3}$ | Delay $\phi_1$ to $\phi_2$ Leading Edges | 80 | | nsec | |
| $t_{DA}$[2] | Address Output Delay From $\phi_2$ | | 200 | nsec | $C_L$ = 100pf |
| $t_{DD}$[2] | Data Output Delay From $\phi_2$ | | 220 | nsec | |
| $t_{DC}$[2] | Signal Output Delay From $\phi_1$, or $\phi_2$ (SYNC, $\overline{WR}$, WAIT, HLDA) | | 120 | nsec | $C_L$ = 50pf |
| $t_{DF}$[2] | DBIN Delay From $\phi_2$ | 25 | 140 | nsec | |
| $t_{DI}$[1] | Delay for Input Bus to Enter Input Mode | | $t_{DF}$ | nsec | |
| $t_{DS1}$ | Data Setup Time During $\phi_1$ and DBIN | 30 | | nsec | |

## TIMING WAVEFORMS [14]

(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 8.0V "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)

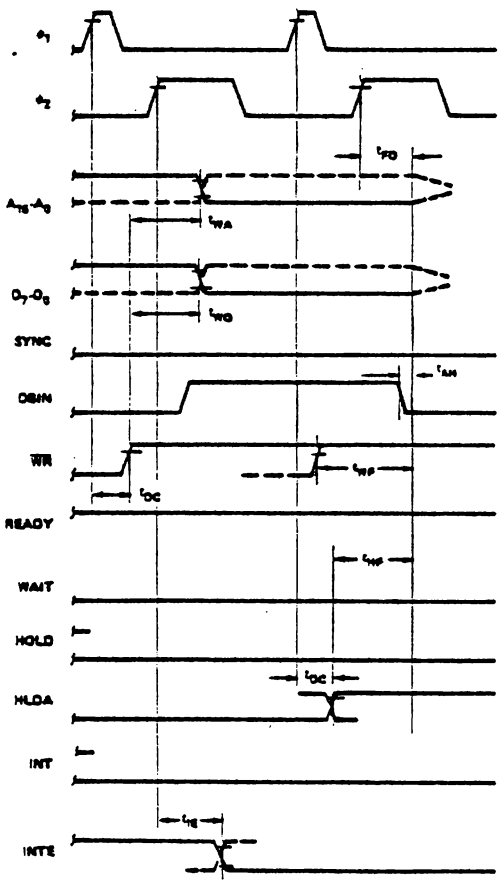



modular micros
martin research

## A.C. CHARACTERISTICS (Continued)

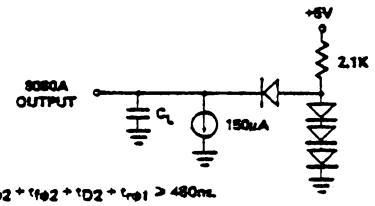
$T_A$ = 0°C to 70°C, $V_{DD}$ = +12V ± 5%, $V_{CC}$ = +5V ± 5%, $V_{BB}$ = -5V ± 5%, $V_{SS}$ = 0V, Unless Otherwise Noted

| Symbol | Parameter | Min. | Max. | Unit | Test Condition |
|---|---|---|---|---|---|
| $t_{DS2}$ | Data Setup Time to $\phi_2$ During DBIN | 150 | | nsec | |
| $t_{DH}$ [1] | Data Hold Time From $\phi_2$ During DBIN | [1] | | nsec | |
| $t_{IE}$ [2] | INTE Output Delay From $\phi_2$ | | 200 | nsec | $C_L$ = 50pf |
| $t_{RS}$ | READY Setup Time During $\phi_2$ | 120 | | nsec | |
| $t_{HS}$ | HOLD Setup Time to $\phi_2$ | 140 | | nsec | |
| $t_{IS}$ | INT Setup Time During $\phi_2$ (During $\phi_1$ in Halt Mode) | 120 | | nsec | |
| $t_H$ | Hold Time From $\phi_2$ (READY, INT, HOLD) | 0 | | nsec | |
| $t_{FD}$ | Delay to Float During Hold (Address and Data Bus) | | 120 | nsec | |
| $t_{AW}$ [2] | Address Stable Prior to $\overline{WR}$ | [5] | | nsec | $C_L$ = 100pf: Address, Data<br>$C_L$ = 50pf: $\overline{WR}$, HLDA, DBIN |
| $t_{DW}$ [2] | Output Data Stable Prior to $\overline{WR}$ | [6] | | nsec | |
| $t_{WD}$ [2] | Output Data Stable From $\overline{WR}$ | [7] | | nsec | |
| $t_{WA}$ [2] | Address Stable From $\overline{WR}$ | [7] | | nsec | |
| $t_{HF}$ [2] | HLDA to Float Delay | [8] | | nsec | |
| $t_{WF}$ [2] | $\overline{WR}$ to Float Delay | [9] | | nsec | |
| $t_{AH}$ [2] | Address Hold Time After DBIN During HLDA | -20 | | nsec | |



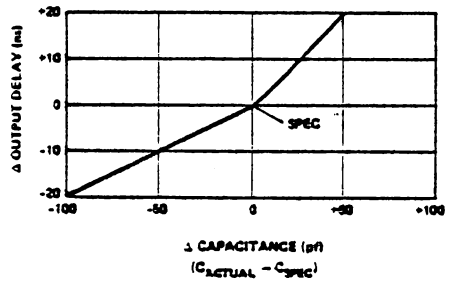

NOTES:

1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured. $t_{DH}$ = 50 ns or $t_{D\beta}$, whichever is less.
2. Load Circuit.




3. $t_{CY} = t_{D3} + t_{r\phi 2} + t_{\phi 2} + t_{f\phi 2} + t_{D2} + t_{r\phi 1} \geq 480$ns.

TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE




4. The following are relevant when interfacing the 8080A to devices having $V_{IH}$ = 3.3V:
   a) Maximum output rise time from .8V to 3.3V = 100ns @ $C_L$ = SPEC.
   b) Output delay when measured to 3.0V = SPEC +60ns @ $C_L$ = SPEC.
   c) If $C_L$ = SPEC, add .8ns/pF if $C_L > C_{SPEC}$, subtract .3ns/pF (from modified delay) if $C_L < C_{SPEC}$.
5. $t_{AW} = 2\,t_{CY} - t_{D3} - t_{r\phi 2} - 140$nsec.
6. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi 2} - 170$nsec.
7. If not HLDA, $t_{WD} = t_{WA} = t_{D3} + t_{r\phi 2} + 10$ns. If HLDA, $t_{WD} = t_{WA} = t_{WF}$.
8. $t_{HF} = t_{D3} + t_{r\phi 2} - 50$ns.
9. $t_{WF} = t_{D3} + t_{r\phi 2} - 10$ns
10. Data in must be stable for this period during DBIN $T_3$. Both $t_{DS1}$ and $t_{DS2}$ must be satisfied.
11. Ready signal must be stable for this period during $T_2$ or $T_W$. (Must be externally synchronized.)
12. Hold signal must be stable for this period during $T_2$ or $T_W$ when entering hold mode, and during $T_3$, $T_4$, $T_5$ and $T_{WH}$ when in hold mode. (External synchronization is not required.)
13. Interrupt signal must be stable during this period of the last clock cycle of any instruction in order to be recognized on the following instruction. (External synchronization is not required.)
14. This timing diagram shows timing relationships only; it does not represent any specific machine cycle.

modular micros martin research

## INSTRUCTION SET

### Summary of Processor Instructions By Alphabetical Order

| Mnemonic | Description | Instruction Code[1] $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ | Clock[2] Cycles |
|---|---|---|---|---|---|---|---|---|---|---|
| ACI | Add immediate to A with carry | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 7 |
| ADC M | Add memory to A with carry | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 7 |
| ADC r | Add register to A with carry | 1 | 0 | 0 | 0 | 1 | S | S | S | 4 |
| ADD M | Add memory to A | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| ADD r | Add register to A | 1 | 0 | 0 | 0 | 0 | S | S | S | 4 |
| ADI | Add immediate to A | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| ANA M | And memory with A | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 7 |
| ANA r | And register with A | 1 | 0 | 1 | 0 | 0 | S | S | S | 4 |
| ANI | And immediate with A | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 7 |
| CALL | Call unconditional | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 17 |
| CC | Call on carry | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 11/17 |
| CM | Call on minus | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 11/17 |
| CMA | Compliment A | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 4 |
| CMC | Compliment carry | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 4 |
| CMP M | Compare memory with A | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 7 |
| CMP r | Compare register with A | 1 | 0 | 1 | 1 | 1 | S | S | S | 4 |
| CNC | Call on no carry | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 11/17 |
| CNZ | Call on no zero | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 11/17 |
| CP | Call on positive | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 11/17 |
| CPE | Call on parity even | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 11/17 |
| CPI | Compare immediate with A | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 7 |
| CPO | Call on parity odd | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 11/17 |
| CZ | Call on zero | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 11/17 |
| DAA | Decimal adjust A | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 4 |
| DAD B | Add B & C to H & L | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 10 |
| DAD D | Add D & E to H & L | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 10 |
| DAD H | Add H & L to H & L | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 10 |
| DAD SP | Add stack pointer to H & L | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 10 |
| DCR M | Decrement memory | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 10 |
| DCR r | Decrement register | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 5 |
| DCX B | Decrement B & C | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 5 |
| DCX D | Decrement D & E | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 5 |
| DCX H | Decrement H & L | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 5 |
| DCX SP | Decrement stack pointer | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 5 |
| DI | Disable Interrupt | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 4 |
| EI | Enable Interrupts | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 4 |
| HLT | Halt | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 7 |
| IN | Input | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 10 |
| INR M | Increment memory | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 10 |
| INR r | Increment register | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 |
| INX B | Increment B & C registers | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 5 |
| INX D | Increment D & E registers | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 5 |
| INX H | Increment H & L registers | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 5 |
| INX SP | Increment stack pointer | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 5 |
| JC | Jump on carry | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 10 |
| JM | Jump on minus | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 10 |
| JMP | Jump unconditional | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 10 |
| JNC | Jump on no carry | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 10 |
| JNZ | Jump on no zero | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 10 |
| JP | Jump on positive | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 10 |
| JPE | Jump on parity even | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 10 |
| JPO | Jump on parity odd | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 10 |
| JZ | Jump on zero | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 10 |
| LDA | Load A direct | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 13 |
| LDAX B | Load A indirect | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 7 |
| LDAX D | Load A indirect | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 7 |
| LHLD | Load H & L direct | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 16 |
| LXI B | Load immediate register Pair B & C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 10 |
| LXI D | Load immediate register Pair D & E | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 10 |
| LXI H | Load immediate register Pair H & L | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 10 |
| LXI SP | Load immediate stack pointer | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 10 |
| MVI M | Move immediate memory | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 10 |
| MVI r | Move immediate register | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| MOV M, r | Move register to memory | 0 | 1 | 1 | 1 | 0 | S | S | S | 7 |
| MOV r, M | Move memory to register | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 7 |
| MOV $_{r1,r2}$ | Move register to register | 0 | 1 | 0 | 0 | 0 | S | S | S | 5 |
| NOP | No-operation | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 |
| ORA M | Or memory with A | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 7 |
| ORA r | Or register with A | 1 | 0 | 1 | 1 | 0 | S | S | S | 4 |
| ORI | Or immediate with A | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 7 |
| OUT | Output | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 10 |
| PCHL | H & L to program counter | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 5 |
| POP B | Pop register pair B & C off stack | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 10 |
| POP D | Pop register pair D & E off stack | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 10 |
| POP H | Pop register pair H & L off stack | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 10 |
| POP PSW | Pop A and Flags off stack | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 10 |
| PUSH B | Push register Pair B & C on stack | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 11 |
| PUSH D | Push register Pair D & E on stack | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 11 |
| PUSH H | Push register Pair H & L on stack | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 11 |
| PUSH PSW | Push A and Flags on stack | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 11 |
| RAL | Rotate A left through carry | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 4 |
| RAR | Rotate A right through carry | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 4 |
| RC | Return on carry | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 5/11 |
| RET | Return | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 10 |
| RLC | Rotate A left | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 4 |
| RM | Return on minus | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 5/11 |
| RNC | Return on no carry | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 5/11 |
| RNZ | Return on no zero | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 5/11 |
| RP | Return on positive | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 5/11 |
| RPE | Return on parity even | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 5/11 |
| RPO | Return on parity odd | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 5/11 |
| RRC | Rotate A right | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 4 |
| RST | Restart | 1 | 1 | A | A | A | 1 | 1 | 1 | 11 |
| RZ | Return on zero | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 5/11 |
| SBB M | Subtract memory from A with borrow | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 7 |
| SBB r | Subtract register from A with borrow | 1 | 0 | 0 | 1 | 1 | S | S | S | 4 |
| SBI | Subtract immediate from A with borrow | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 7 |
| SHLD | Store H & L direct | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 16 |
| SPHL | H & L to stack pointer | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 5 |
| STA | Store A direct | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 13 |
| STAX B | Store A indirect | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 7 |
| STAX D | Store A indirect | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 7 |
| STC | Set carry | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 4 |
| SUB M | Subtract memory from A | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 7 |
| SUB r | Subtract register from A | 1 | 0 | 0 | 1 | 0 | S | S | S | 4 |
| SUI | Subtract immediate from A | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 7 |
| XCHG | Exchange D & E, H & L Registers | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 4 |
| XRA M | Exclusive Or memory with A | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 7 |
| XRA r | Exclusive Or register with A | 1 | 0 | 1 | 0 | 1 | S | S | S | 4 |
| XRI | Exclusive Or immediate with A | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 7 |
| XTHL | Exchange top of stack, H & L | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 18 |

NOTES: 1. DDD or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

modular micros | martin research

# 4. The Modular Microcomputer System

*REV. B*

The minimal *MIKE 3* system is sold with three boards: a Model 471 (or 431) CPU board, with 8080; a Model 420 Console Board, with a 20-key keyboard and six fully-decoded digits; and a Model 423 PROM/RAM with 512 bytes of RAM (expandable to 1K), and 256 bytes of PROM (expandable to 2K).

The PROM provided with the *MIKE 3* system allows the user to store programs in RAM using the calculator-type keyboard. Instructions are entered *either* in octal *or* in hexadecimal format. (The purchaser's choice should be specified; octal is shipped when no preference is expressed.) Both instructions and data are displayed automatically, in fully decoded digital form, on the six console digits. The user can rapidly address any location in memory, and display the contents at that address. He can increment or decrement the memory location, using the keyboard, and can execute a program in memory starting at any selected address.

The *MIKE 3* Monitor supports software debugging with *program breakpoints*. The user inserts special one-byte RESTART instructions in the program; the computer automatically stops when a breakpoint is encountered, and the contents of all CPU registers and flags may be inspected successively on the LED digits, under control of the keyboard.

A useful option is the *MIKE 3 DEMO PROM*, which provides programs for testing the keyboard, display, and RAM memory at any location; a hexadecimal-to-decimal conversion routine; and software for causing the display to operate as a 24-hour digital clock (hours, minutes, seconds).

An option useful for the hardware designer is the Model 425 breadboard. A blank board of standard *modular micro* dimensions, the 425 contains over nine hundred pre-tinned holes in convenient topology for standard DIP ICs. All important control signals within the system are available for interfacing with custom circuitry. The architecture of the computer makes the addition of input and output ports very simple.

In order to expand memory, extra RAM chips may be added to the PROM/RAM board, as may extra PROMs as well. The Model 405 RAM board satisfies the need for large memory capacity, with a full 4K bytes of high-speed static RAM (450 ns access time).

Other options--both hardware and software--are currently under development. For details, stay in touch with Martin Research!

*FOR INFORMATION ON THE MODEL 814 DEBUG SYSTEM, SEE OP814, IN THE OPERATING SECTION OF THIS MANUAL.*

| PIN NO. | SIGNAL | DIRECTION | TYPE | COMMENTS |
|---|---|---|---|---|
| 1 | DB7 | ↔ | 3S | Data bus, high-order bit |
| 2 | DB6 | ↔ | 3S | ↓ |
| 3 | DB5 | ↔ | 3S | ↓ |
| 4 | DB4 | ↔ | 3S | ↓ |
| 5 | DB3 | ↔ | 3S | ↓ |
| 6 | DB2 | ↔ | 3S | ↓ |
| 7 | DB1 | ↔ | 3S | ↓ |
| 8 | DB0 | ↔ | 3S | Data bus, low-order bit |
| 9 | DH7 | → | TTL | Address bus *(Note: DH7 and DH6 are* |
| 10 | DH6 | → | TTL | ↓ *grounded in 8008 system)* |
| 11 | DH5 | → | TTL | ↓ |
| 12 | DH4 | → | TTL | ↓ |
| 13 | DH3 | → | TTL | ↓ |
| 14 | DH2 | → | TTL | ↓ |
| 15 | DH1 | → | TTL | ↓ |
| 16 | DH0 | → | TTL | High-order address, low-order bit |
| 17 | DL7 | → | TTL | Low-order address, high-order bit |
| 18 | DL6 | → | TTL | ↓ |
| 19 | DL5 | → | TTL | ↓ |
| 20 | DL4 | → | TTL | ↓ |
| 21 | DL3 | → | TTL | ↓ |
| 22 | DL2 | → | TTL | ↓ |
| 23 | DL1 | → | TTL | ↓ |
| 24 | DL0 | → | TTL | Low-order address, low-order bit |
| 25 | | | | *Spare* |
| 26 | $\overline{\text{ADV}}$ | → | TTL | Address valid |
| 27 | $\overline{\text{IOR}}$, $\overline{\text{DINB}}$ | → | TTL | Input strobe signal |
| 28 | $\overline{\text{DIN}}$ | → | TTL | Input strobe signal |
| 29 | $\overline{\text{PCW}}$ | → | TTL | Memory write cycle signal |
| 30 | $\overline{\text{MEMR}}$ | → | TTL | Memory read strobe |
| 31 | $\overline{\text{IOW}}$ | → | TTL | I/O write signal |
| 32 | $\overline{\text{BSY}}$ | ← | O.C. | Wait signal (slow memory/I/O) |
| 33 | $\overline{\text{INTS}}$ | → | TTL | Interrupt strobe |
| 34 | $\overline{\text{INTR}}$ | ← | O.C. | Interrupt request |
| 35 | $\overline{\text{STEW}}$ | → | TTL | Memory write (=$\overline{\text{MEMW}}$) |
| 36 | $\overline{\text{MEM}}$ | → | TTL | Memory enable signal |
| 37 | $\overline{\text{HLDR}}$ | ← | O.C. | Hold request |
| 38 | $\overline{\text{HACK}}$ | → | TTL | Hold acknowledge |
| 39 | GND | | *Power* | |
| 40 | GND | | *Power* | |
| 41 | GND | | *Power* | |
| 42 | GND | | *Power* | |
| 43 | +12 V | | *Power* | |
| 44 | +12 V | | *Power* | |
| 45 | -12 V | | *Power* | |
| 46 | -9 V | | *Power* | |
| 47 | -9 V | | *Power* | |
| 48 | +5 V | | *Power* | |
| 49 | +5 V | | *Power* | |
| 50 | +5 V | | *Power* | |

# 5. Operating Instructions

REV. B

The *MIKE 3* microcomputer is supplied with a Console Board, featuring a calculator-style keyboard with twenty keys, and a six-digit display. The keyboard is used to program the microcomputer, and the display is used to visualize the results.

The keyboard's twenty keys include the digits 0 through 7 and a number of special-purpose keys. Each of these special keys is assigned a function useful in programming the microcomputer. A table of these functions appears on a following page.

In order to introduce the user to the *MIKE 3*, we will introduce the keys on the keyboard, one by one. We start by assuming that the assembled *MIKE 3* is connected to a suitable power supply; the power is turned on; the six-digit display glows red, probably displaying some random collection of the digits 0 through 9 and A through F; and the user is sitting in front of the micro with a puzzled expression.

> *Some of the more technical details on microcomputer operations will be described in indented paragraphs in italic type, like this. The beginner may choose to skip these sections for now.*

## SEC. 1 STARTING THE MICRO WITH THE INTERRUPT KEYS

In order to start up the microcomputer, it is necessary to provide it with an *interrupt*. Two keys on the Model 420 Console Board are associated with the interrupt function: T and -N. In order to interrupt the system, it is necessary to press first the *PRESET INTERRUPT (T)* key, and then the *INTERRUPT (-N)* key. The reader should try this.

The display now begins to alternate between two messages:

014 000 - CCCxyz - 014000 - CCCxyz (etc.)

where xyz is a set of three digits with random values between 0 and 7. Once the display begins to blink in this manner, the microcomputer has been started and is ready to be programmed.

> *The PRESET INTERRUPT (T) key sets a flip-flop on the 420 Console Board, which causes the Interrupt Request (INTR) line on the microcomputer bus to go low. The INTERRUPT (-N) key resets this flip-flop, causing the INTR line to go high again. As this line goes high, the 8080 microprocessor recognizes an interrupt request. Circuitry on the CPU board automatically requests a level 1 interrupt from the 8080. If interrupts are not masked, location 010 in the 1702A PROM on the PROM/RAM board is read to obtain the next instruction. This gives control to the M3M MONITOR program, which takes over. If interrupts are masked, the CPU board finds the interrupt was not serviced $2^7$ clock pulses later, so it causes a Reset.*
>
> *The number 014000 is an address in RAM memory; xyz is the contents of memory at that address. The top of the push down stack is 013377.*

REV. B

SEC. 1 STARTING THE MICRO WITH THE INTERRUPT KEYS (CONTINUED)

*The complete stack can be read using the READ key. The full significance of these numbers is explained in Sec. 5.*

SEC. 2 SETTING UP AN ADDRESS TO STORE A PROGRAM IN MEMORY

In order to make the microcomputer execute a program, the user must generally write a series of instructions into memory and then execute them. The first step is to choose a suitable block of RAM into which to write the program. In the basic *MIKE 3* system, there are 512 (decimal) bytes (eight-bit words) of RAM, located in pages 010 (010 000 through 010 377) and 013 (013 000 through 013 377). In preparation for storing a program, we will make the microcomputer point to location 010 000.

We assume that the microcomputer has received its initial interrupt and is still blinking the messages 014000 and CCCxyz.

First, set up the high-order memory address, that is, 010. Punch 1, 0. (It is unnecessary to enter leading zeros.) The display will read AAA010. Punch the *LOAD H* (×) key. Now the display shows this new memory location, 010 000, and its contents.

*The Monitor program shifts digits to the left as new ones are entered, as on a handheld calculator. As a digit is being entered into the display, it is also being stored in the 8080 E register. The LOAD key causes the E data to move to the H. Then the E is cleared to zero, and the display resumes blinking memory address and contents.*

Second, set up the low-order memory address, that is, 000. Since it is unnecessary to enter leading zeros, simply punch *LOAD L* (÷).

*The E register was cleared to zero at the end of the preceding operation, LOAD H. The Monitor loads the 8080 L register from the E.*

Now the display will begin to alternate between two messages:

010000 - CCCxyz - 010000 - CCCxyz (etc.)

--where the xyz once again is some random number. The meaning: we are looking at location 010 000 in memory; its present contents are xyz. This is the first location in the *MIKE 3's* array of RAM, and we are now ready to load in a program.

SEC. 3 A SHORT TEST PROGRAM

Here is a short program for testing the *MIKE 3*.

SEC. 3 A SHORT TEST PROGRAM (CONTINUED)

| *LOCATION* | *OCTAL* | *MNEMONIC* | *COMMENTS* |
|---|---|---|---|
| 010 000 | 173 | MOV A,E | LOAD A REGISTER FROM E REGISTER |
| 010 001 | 306 021 | ADI 021 | ADD 021, I.E., 00 010 001 IN BINARY, OR REGROUPING, 0001 0001 |
| 010 003 | 322 007 010 | JNC 010 007 | IF THE A REGISTER HAS NOT YET OVERFLOWED, SKIP THE NEXT INSTRUCTION |
| 010 006 | 257 | XRA A | CLEAR A REGISTER |
| 010 007 | 137 | MOV E,A | STORE THE COUNT IN THE E REGISTER |
| 010 010 | 026 333 | MVI D, 333( | OUTPUT TO DIGITS 0, 1 |
| 010 012 | 022 | STAX D ( | |
| 010 013 | 026 335 | MVI D, 335( | OUTPUT TO DIGITS 2, 3 |
| 010 015 | 022 | STAX D ( | |
| 010 016 | 026 337 | MVI D, 337( | OUTPUT TO DIGITS 4, 5 |
| 010 020 | 022 | STAX D ( | |
| 010 021 | 347 | RST 040 | WAIT A SECOND |
| 010 022 | 303 000 010 | JMP 010 000 | START AGAIN |

The steps for loading the above program are listed below:

1. Punch 173 (display reads AAA173).
2. Punch the *WRITE* (±) key.

   *This key causes the 8080 to write the value contained in its E register, into memory, at the location defined by the H and L registers - i.e., 010 000.*

   The display should now alternate between the readings:

   010001 - CCCxyz - 010001 - CCCxyz (etc.)

   Notice that, after the *WRITE* key is pressed, the *MONITOR* program proceeds to the next memory location.

3. Punch 306 (display reads AAA306). Now punch the *WRITE* key. This loads the instruction 306 into location 010 001, and the *MONITOR* now steps on to the next location in memory, 010 002.

4. 21, WRITE
5. 322, WRITE
6. 7, WRITE
7. 10, WRITE
8. 257, WRITE
9. 137, WRITE
10. 26, WRITE
11. 333, WRITE
12. 32, WRITE
13. 26, WRITE
14. 335, WRITE
15. 32, WRITE
16. 26, WRITE
17. 337, WRITE
18. 32, WRITE
19. 347, WRITE
20. 303, WRITE
21. WRITE
22. 10, WRITE

18. At this point, the whole program should have been loaded properly into memory. Let's check. Punch the *LOAD L* (÷) key once again to return the memory address to 010 000. The display should now alternate between 010000 and CCC173-- the 173 being the instruction we entered in Step 1, above.

SEC. 3 A SHORT TEST PROGRAM (CONTINUED)

19. Now push the *READ* (-) key. The display should now move to the next location in memory and alternate between 010001 and CCC306. This was the second instruction entered. Just as with the *WRITE* key, the *READ* key automatically goes to the next location in memory-- but without changing what is stored there.

20. Keep pressing the *READ* (-) key, checking the contents of memory through location 010 024.

21. If there is a mistake, enter the last three digits of the memory address at which the mistake occurred, using the number keys. Now push the *LOAD L* (÷) key to set the memory pointer back to that location. Enter the correct instruction using the number keys, and push the *WRITE* (±) to write the correct value into memory.

22. Now execute the program, using the *EXECUTE* key (•). The display should proceed to count through the sixteen hexadecimal numbers, all digits at once, changing about once every 1.3 seconds:

    111111 - 222222 - 333333 - 444444 - 555555 - 666666 - 777777 -
    888888 - 999999 - AAAAAA - bbbbbb - CCCCCC - dddddd - EEEEEE -
    FFFFFF - 000000 - 111111 - 222222, etc.

SEC. 4 8 AND 9 KEY FUNCTIONS

The keys 8 and 9 are assigned special functions by the *MIKE 3 MONITOR*. Since the keyboard is read in octal format, only the digits 0 through 7 are normally used in entering data or memory addresses into the computer. The Monitor interprets the 8 as a command to recall the stack pointer to the H, L register pair.

The 9 key causes the microcomputer to execute a JUMP instruction to memory location 001 000. This is the first location of the second page in memory, a convenient place for the user to install a second PROM. For example, the optional *MIKE 3 Demo PROM* is usually inserted here. If no PROM has been installed at this location, the computer will usually hit a *RESTART* instruction and return to the Monitor.

SEC. 5 CONTINUE KEY: HANDLING INTERRUPTS

The interrupt feature of the microprocessor is used not only to start the computer, but to *jam* a high-priority program routine into the middle of the main line program. After the interrupt routine is over, the main line program continues where it left off.

When the *MIKE 3* receives an interrupt, the Monitor program will save register and flag PSW (processor status word) status in the push down stack as is shown in the following chart.

REV. B

SEC. 5 CONTINUE KEY: HANDLING INTERRUPTS (CONTINUED)

| *GENERAL STACK POSITIONS* | *REGISTERS* |
|---|---|
| Stack Pointer + 11 | PC - Old MSB |
| " + 10 | PC - Old LSB |
| " + 7 | A* |
| " + 6 | PSW* |
| " + 5 | B |
| " + 4 | C |
| " + 3 | D |
| " + 2 | E |
| " + 1 | H |
| Stack Pointer | L |

**Note: the A and PSW words are reversed in order when using the uPD8080 from NEC Microcomputers*

The user can try this out by executing the sample program in Section 3 and interrupting the microcomputer while the program is operating (press T, -N). The digits on the Console board will now start blinking at 013 360. The contents of the L register, at the time when the interrupt request was received by the 8080, are located here. By successively pushing the *READ* key, the user can inspect the other locations, through 013377. In general, stack addresses 013 372 through 013 377 will contain the Program Counter addresses associated with the three CAL or PUSH instructions used by the 1.3 second time-out subroutine. Therefore, the typical stack positions of the CPU registers are:

| *GENERAL STACK POSITIONS* | *REGISTERS* | *TYPICAL STACK POSITIONS* |
|---|---|---|
| Stack Pointer + 20 | Not Used | 014 000 |
| " + 17 | ⎤ | 013 377 |
| " + 16 | ⎥ Sub-routine | 013 376 |
| " + 15 | ⎥ addresses being | 013 375 |
| " + 14 | ⎥ used by Monitor at | 013 374 |
| " + 13 | ⎥ time of interrupt | 013 373 |
| " + 12 | ⎦ | 013 372 |
| " + 11 | PC - Old MSB | 013 371 |
| " + 10 | PC - Old LSB | 013 370 |
| " + 7 | A* | 013 367 |
| " + 6 | PSW* | 013 366 |
| " + 5 | B | 013 365 |
| " + 4 | C | 013 364 |
| " + 3 | D | 013 363 |
| " + 2 | E | 013 362 |
| " + 1 | H | 013 361 |
| Stack Pointer | L | 013 360 |

**Note: A and PSW reversed in order in uPD 8080 from NEC Microcomputers*

The PSW includes Zero (Z), Sign (S), Parity (P), Carry (CY) and Auxiliary Carry (carry between bits 3 and 4), and is stored in the eight-bit memory word as shown on the following page.

SEC. 5 CONTINUE KEY: HANDLING INTERRUPTS (CONTINUED)

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| S | Z | Ø | AC | Ø | P | 1 | CY |

*Note: in the uPD 8080 from NEC Microcomputers, D5 is an extra flag, SUB, allowing for decimal adjust after subtraction; D3 is a 1.*

By definition of the functions of these flags, there are only 20 valid combinations of the five flag bits. These are shown in the chart below. The value of the PSW (processor status word) is in octal.

| PSW | S | Z | AC | P | CY |
|---|---|---|---|---|---|
| 002 | | | | | |
| 003 | | | | | ✓ |
| 006 | | | | ✓ | |
| 007 | | | | ✓ | ✓ |
| 022 | | | ✓ | | |
| 023 | | | ✓ | | ✓ |
| 026 | | | ✓ | ✓ | |
| 027 | | | ✓ | ✓ | ✓ |
| 106 | | ✓ | | ✓ | |
| 107 | | ✓ | | ✓ | ✓ |
| 126 | | ✓ | ✓ | ✓ | |
| 127 | | ✓ | ✓ | ✓ | ✓ |
| 202 | ✓ | | | | |
| 203 | ✓ | | | | ✓ |
| 206 | ✓ | | | ✓ | |
| 207 | ✓ | | | ✓ | ✓ |
| 222 | ✓ | | ✓ | | |
| 223 | ✓ | | ✓ | | ✓ |
| 226 | ✓ | | ✓ | ✓ | |
| 227 | ✓ | | ✓ | ✓ | ✓ |

S = Sign
Z = Zero
AC = Auxiliary carry
P = Parity
CY = Carry

A check mark (✓) in the table to the left means that the designated flag is *set*--ie, logic high (a 1).

*The table to the left must be modified slightly for the extra SUB flag, when using the uPD 8080 from NEC.*

The table below can be copied and filled in by the user while using the status-saving feature of the *MIKE 3*.

| | TRIAL #1 | TRIAL #2 | TRIAL #3 | TRIAL #4 | TRIAL #5 |
|---|---|---|---|---|---|
| PCH | | | | | |
| PCL | | | | | |
| PCH | | | | | |
| PCL | | | | | |
| PCH | | | | | |
| PCL | | | | | |
| A | | | | | |
| PSW | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |
| E | | | | | |
| H | | | | | |
| L | | | | | |

modular micros ⇄ martin research

REV. B

SEC. 5 CONTINUE KEY: HANDLING INTERRUPTS (CONTINUED)

After having interrupted the counting program and inspected the saved status, press the *CONTINUE* (C) key. The counting program should resume where it left off.

*NOTE:* If the counting program is interrupted more than once, the *CONTINUE* key must be pressed the same number of times to allow resumption of the counting program. Each interrupt successively saves register and flag status, pushing deeper onto the stack. Each continue operation restores registers and flags, popping back up the stack. Specifically: as discussed above, the stack pointer is at 014 000 before any interrupts are received (and this address is displayed on the Console board). After one interrupt, the stack pointer typically reads 013 360. After two interrupts, it is at 013 340. Pressing the *CONTINUE* key will now restore register status (pop up the stack) and the pointer will again read 013 360. Pressing the key again will restore the pointer to its initial reading, 014 000.

SEC. 6 CONTROL KEYS, *M* AND *=K*

The two Console board control keys, M and =K, are analogous to shift keys on a typewriter. Pressed at the same time as some other key on the keyboard, they can be used to create modes different from the basic interpretation of that key. These keys are used in the *MIKE 3* Monitor program (and in the monitor programs of other computers in the *modular micro* family). (For a circuit description of the 420, see the data sheet.)

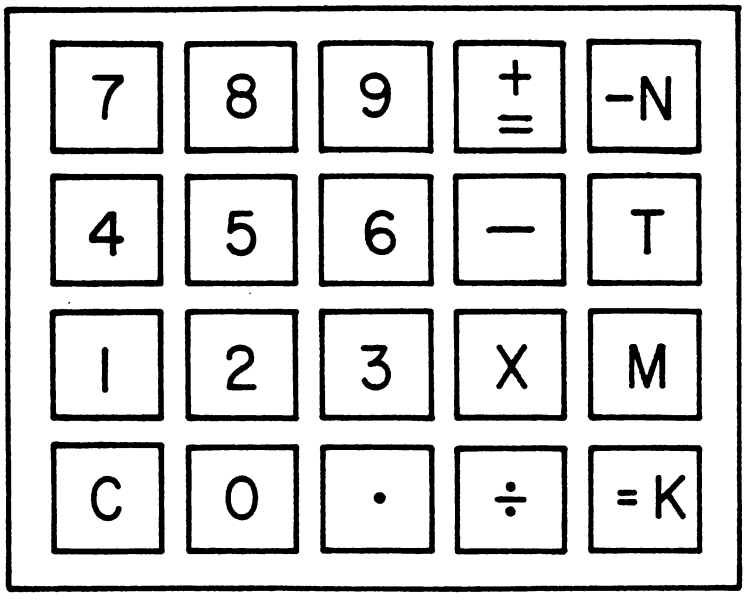
The following two pages illustrate the functions of the 420 Console board keys for two different *MIKE 3* Monitor programs--the octal and hexadecimal versions.

SEC. 6 CONTROL KEYS, *M* AND *=K (cont'd)*

*REV. C*

*MIKE 3 MONITOR, OCTAL VERSION*

## KEYBOARD FUNCTIONS




| | *NO SHIFT KEYS* | |
|---|---|---|
| KEY | HEX | FUNCTION |
| 0 | *00* | 0 |
| 1 | *01* | 1 |
| 2 | *02* | 2 |
| 3 | *03* | 3 |
| 4 | *04* | 4 |
| 5 | *05* | 5 |
| 6 | *06* | 6 |
| 7 | *07* | 7 |
| 8 | *08* | Set H,L to SP |
| 9 | *09* | Execute @ 001 000 |
| ± | *0A* | Write; inc.; read |
| - | *0b* | Increment & read |
| × | *0C* | Load H |
| ÷ | *0d* | Load L |
| · | *0E* | Execute @ 010 000 |
| C | *0F* | Continue |

| | M *KEY PRESSED* | | =K *KEY PRESSED* | | M *AND* =K *KEYS PRESSED* | |
|---|---|---|---|---|---|---|
| KEY | HEX | FUNCTION | HEX | FUNCTION | HEX | FUNCTION |
| 0 | *10* | SP =HL =014 000 | *20* | | *30* | |
| 1 | *11* | Execute at H,L | *21* | | *31* | |
| 2 | *12* | Set SP = H,L | *22* | | *32* | |
| 3 | *13* | Decrement & read | *23* | | *33* | |
| 4 | *14* | | *24* | | *34* | |
| 5 | *15* | | *25* | | *35* | |
| 6 | *16* | | *26* | | *36* | |
| 7 | *17* | | *27* | | *37* | |
| 8 | *18* | | *28* | | *38* | |
| 9 | *19* | | *29* | | *39* | |
| ± | *1A* | | *2A* | | *3A* | |
| - | *1b* | Decrement & read | *2b* | | *3b* | |
| × | *1C* | | *2C* | | *3C* | |
| ÷ | *1d* | | *2d* | | *3d* | |
| · | *1E* | | *2E* | | *3E* | |
| C | *1F* | | *2F* | | *3F* | |

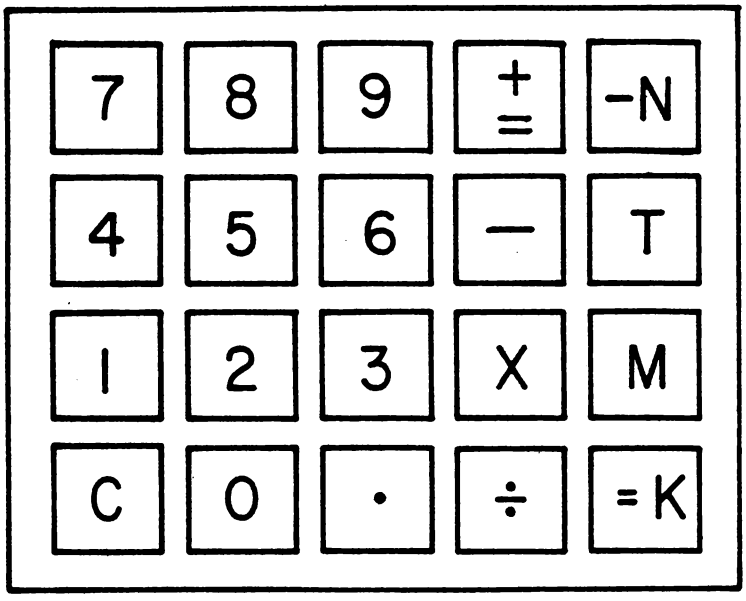
| | |
|---|---|
| -N | *PRESET INTERRUPT* |
| T | *INTERRUPT* |

SEC. 6 CONTROL KEYS, *M* AND *=K (cont'd)*

REV. D

*MIKE 3 MONITOR, HEXADECIMAL VERSION*

## KEYBOARD FUNCTIONS




| | *NO SHIFT KEYS* | |
|---|---|---|
| KEY | HEX | FUNCTION |
| 0 | *00* | 0 |
| 1 | *01* | 1 |
| 2 | *02* | 2 |
| 3 | *03* | 3 |
| 4 | *04* | 4 |
| 5 | *05* | 5 |
| 6 | *06* | 6 |
| 7 | *07* | 7 |
| 8 | *08* | 8 |
| 9 | *09* | 9 |
| ± | *0A* | A |
| - | *0b* | B |
| × | *0C* | C |
| ÷ | *0d* | D |
| · | *0E* | E |
| C | *0F* | F |

| | M *KEY PRESSED* | | =K *KEY PRESSED* | | M *AND* =K *KEYS PRESSED* | |
|---|---|---|---|---|---|---|
| KEY | HEX | FUNCTION | HEX | FUNCTION | HEX | FUNCTION |
| 0 | *10* | SP =HL = 0C 00 | *20* | | *30* | |
| 1 | *11* | Execute at H,L | *21* | | *31* | |
| 2 | *12* | Set SP = H,L | *22* | | *32* | |
| 3 | *13* | | *23* | | *33* | |
| 4 | *14* | | *24* | | *34* | |
| 5 | *15* | | *25* | | *35* | |
| 6 | *16* | Decrement & read | *26* | | *36* | |
| 7 | *17* | | *27* | | *37* | |
| 8 | *18* | Set H,L to SP | *28* | | *38* | |
| 9 | *19* | Execute @ 01 00 | *29* | | *39* | |
| ± | *1A* | Write; inc; read | *2A* | | *3A* | |
| - | *1b* | Increment & read | *2b* | Decrement & read | *3b* | |
| × | *1C* | Load H | *2C* | | *3C* | |
| ÷ | *1d* | Load L | *2d* | | *3d* | |
| · | *1E* | Execute @ 08 00 | *2E* | | *3E* | |
| C | *1F* | Continue | *2F* | | *3F* | |

| | |
|---|---|
| -N | *PRESET INTERRUPT* |
| T | *INTERRUPT* |

# 8080 INSTRUCTION SET

Summary of Processor Instructions in Alphabetical Order

| *Mnemonic* | *Description* | *Octal* | *Code* | |
|---|---|---|---|---|
| ACI *data* | Add immediate to A with carry | 3 | 1 | 6 |
| ADC *r* | Add register to A with carry | 2 | 1 | *r* |
| ADD *r* | Add register to A | 2 | 0 | *r* |
| ADI *data* | Add immediate to A | 3 | 0 | 6 |
| ANA *r* | AND register with A | 2 | 4 | *r* |
| ANI *data* | AND immediate with A | 3 | 4 | 6 |
| CALL *addr* | Call unconditional | 3 | 1 | 5 |
| C*c* *addr* | Call on condition | 3 | *c* | 4 |
| CMA | Complement A | 0 | 5 | 7 |
| CMC | Complement carry | 0 | 7 | 7 |
| CMP *r* | Compare register with A | 2 | 7 | *r* |
| CPI *data* | Compare immediate with A | 3 | 7 | 6 |
| DAA | Decimal adjust A | 0 | 4 | 7 |
| DAD *rp* | Add register pair to HL | 0 | *rp+1* | 1 |
| DCR *r* | Decrement register | 0 | *r* | 5 |
| DCX *rp* | Decrement register pair | 0 | *rp+1* | 3 |
| DI | Disable interrupts | 3 | 6 | 3 |
| EI | Enable interrupts | 3 | 7 | 3 |
| HLT | Halt | 1 | 6 | 6 |
| IN *port* | Input | 3 | 3 | 3 |
| INR *r* | Increment register | 0 | *r* | 4 |
| INX *rp* | Increment register pair | 0 | *rp* | 3 |
| JMP *addr* | Jump unconditional | 3 | 0 | 3 |
| J*c* *addr* | Jump on condition | 3 | *c* | 2 |
| LDA *addr* | Load A direct | 0 | 7 | 2 |
| LDAX *rp* | Load A indirect *(only B, D)* | 0 | *rp+1* | 2 |
| LHLD *addr* | Load HL direct | 0 | 5 | 2 |
| LXI *rp, data* | Load immediate register pair | 0 | *rp* | 1 |
| MOV *d, s* | Move register to register | 1 | *d* | *s* |
| MVI *r, data* | Move immediate register | 0 | *r* | 6 |
| NOP | No operation | 0 | 0 | 0 |

| *Mnemonic* | *Description* | *Octal* | *Code* | |
|---|---|---|---|---|
| ORA *r* | OR register with A | 2 | 6 | *r* |
| ORI *data* | OR immediate with A | 3 | 6 | 6 |
| OUT *port* | Output | 3 | 2 | 3 |
| PCHL | HL to program counter | 3 | 5 | 1 |
| POP *rp* | Pop register pair off stack *(only B, D, H)* | 3 | *rp* | 1 |
| POP PSW | Pop A and flags off stack | 3 | 6 | 1 |
| PUSH *rp* | Push register pair onto stack *(only B, D, H)* | 3 | *rp* | 5 |
| PUSH PSW | Push A and flags onto stack | 3 | 6 | 5 |
| RAL | Rotate A left through carry | 0 | 2 | 7 |
| RAR | Rotate A right through carry | 0 | 3 | 7 |
| R*c* | Return on condition | 3 | *c* | 0 |
| RET | Return | 3 | 1 | 1 |
| RLC | Rotate A left | 0 | 0 | 7 |
| RRC | Rotate A right | 0 | 1 | 7 |
| RST *n* | Restart | 3 | *n* | 7 |
| SBB *r* | Subtract reg. from A w/borrow | 2 | 3 | *r* |
| SBI *data* | Subtract imm. from A w/borrow | 3 | 3 | 6 |
| SHLD | Store HL direct | 0 | 4 | 2 |
| SPHL | HL to stack pointer | 3 | 7 | 1 |
| STA *addr* | Store A direct | 0 | 6 | 2 |
| STAX *rp* | Store A indirect *(only B, D)* | 0 | *rp* | 2 |
| STC | Set carry | 0 | 6 | 7 |
| SUB *r* | Subtract register from A | 2 | 2 | *r* |
| SUI *data* | Subtract immediate from A | 3 | 2 | 6 |
| XCHG | Exchange DE, HL register pairs | 3 | 5 | 3 |
| XRA *r* | Exclusive OR register with A | 2 | 5 | *r* |
| XRI *data* | Exclusive OR immediate with A | 3 | 5 | 6 |
| XTHL | Exchange top of stack with HL | 3 | 4 | 3 |

*ABBREVIATIONS*

| *Abbrev.* | *Description* | *Bits* |
|---|---|---|
| *addr* | Memory address | 16 |
| *c* | Condition | 3 |
| *d* | Destination | 3 |
| *data* | Data | 8/16 |
| *port* | I/O port | 8 |
| *r* | Register | 3 |
| *rp* | Register pair | 2 |
| *s* | Source | 3 |

*INSTRUCTION CODE VALUES*

| *OCTAL CODE* | *REGISTER (d, r, s)* | *REGISTER PAIR (rp)* | *REGISTER PAIR (rp+1)* | *CONDITION (c) Abbrev.* | *CONDITION (c) Description* | *CONDITION (c) Flag* |
|---|---|---|---|---|---|---|
| 0 | B | BC | -- | NZ | Not zero | Z = 0 |
| 1 | C | -- | BC | Z | Zero | Z = 1 |
| 2 | D | DE | -- | NC | No carry | CY = 0 |
| 3 | E | -- | DE | C | Carry | CY = 1 |
| 4 | H | HL | -- | PO | Parity odd | P = 0 |
| 5 | L | -- | HL | PE | Parity even | P = 1 |
| 6 | M | SP | -- | P | Plus | S = 0 |
| 7 | A | -- | SP | M | Minus | S = 1 |

*8080 INSTRUCTIONS IN OCTAL FORMAT*

The 8080 is an 8-bit microprocessor, and its instruction code tends to break down into a 2/3/3-bit grouping. This is why it was convenient in this chart to list 8080 instructions in octal format, where digits follow this same pattern.

Hexadecimal notation (4/4-bit grouping) is often used instead of octal, but since hex does not correspond directly to the way in which the 8080 is microcoded, hex instruction codes are not readily listed in condensed graphic form, and are more difficult for the programmer to learn.

*Copyright © 1976, Martin Research*

# 8080 INSTRUCTION SET

## REGISTER TRANSFERS

MOV B, H = MOVE TO THE B REG. THE CONTENTS OF THE H REGISTER.

A = 7 C = 1 E = 3 L = 5
B = 0 D = 2 H = 4 M = 6

| | | DESTINATION A | B | C | D | E | H | L | M |
|---|---|---|---|---|---|---|---|---|---|
| SOURCE | A | 177 | 107 | 117 | 127 | 137 | 147 | 157 | 167 |
| | B | 170 | 100 | 110 | 120 | 130 | 140 | 150 | 160 |
| | C | 171 | 101 | 111 | 121 | 131 | 141 | 151 | 161 |
| | D | 172 | 102 | 112 | 122 | 132 | 142 | 152 | 162 |
| | E | 173 | 103 | 113 | 123 | 133 | 143 | 153 | 163 |
| | H | 174 | 104 | 114 | 124 | 134 | 144 | 154 | 164 |
| | L | 175 | 105 | 115 | 125 | 135 | 145 | 155 | 165 |
| | M | 176 | 106 | 116 | 126 | 136 | 146 | 156 | — |
| MOVE IMMEDIATE | MVI r | 076 | 006 | 016 | 026 | 036 | 046 | 056 | 066 |

## DOUBLE-PRECISION TRANSFERS

| SOURCE | DESTINATION B, C | D, E | H, L | SP | PC |
|---|---|---|---|---|---|
| D, E | —— | —— | XCHG 353 | —— | —— |
| H, L | —— | XCHG 353 | —— | SPHL 371 | PCHL 351 |
| IMMED. | LXI B 001 | LXI D 021 | LXI H 041 | LXI SP 061 | —— |

## INDIRECT TRANSFERS

| BYTES | INSTR. | REG. | DATA | MEMORY ADDR. BY: |
|---|---|---|---|---|
| 1 | STAX B 002 | A REGISTER | → | B, C |
| | LDAX B 012 | | ← | |
| | STAX D 022 | | → | D, E |
| | LDAX D 032 | | ← | |
| | MOV M, r SEE ABOVE | INDEX REG. r | → | H, L |
| | MOV r, M SEE ABOVE | | ← | |
| | MVI M 066 | 2ND BYTE OF INST. | → | |
| 2 | PUSH B 305 | B, C | → | SP (Stack Pointer) PUSH DECR. SP BY 2 |
| | POP B 301 | | ← | |
| | PUSH D 325 | D, E | → | |
| | POP D 321 | | ← | POP INCR. SP BY 2 |
| | PUSH H 345 | H, L | → | |
| | POP H 341 | | ← | |
| | PUSH PSW 365 | A, PSW | → | |
| | POP PSW 361 | | ← | |
| | XTHL 343 | H, L | ↔ | SP |

## DIRECT LOAD & STORE

| BYTES | INSTR. | REG. | DATA | MEMORY ADDR. BY: |
|---|---|---|---|---|
| 1 | STA 062 | A | → | 2nd AND 3rd BYTES OF INSTR. |
| | LDA 072 | | ← | |
| 2 | SHLD 042 | H, L | → | |
| | LHLD 052 | | ← | |

## INPUT/OUTPUT

| INPUT | OUTPUT |
|---|---|
| IN 333 | OUT 323 |
| 2ND BYTE = I/O ADDR. | |

| UNUSED | |
|---|---|
| 010 | 313 |
| 020 | 331 |
| 030 | 336 |
| 040 | 356 |
| 050 | 375 |
| 060 | — |
| 070 | — |

Copyright © 1976
MARTIN RESEARCH
ALL RIGHTS RESERVED

## PROGRAM CONTROL

| | UNCONDITIONAL | INDIRECT | ZERO FALSE | ZERO TRUE | CARRY FALSE | CARRY TRUE | PARITY FALSE | PARITY TRUE | SIGN FALSE | SIGN TRUE |
|---|---|---|---|---|---|---|---|---|---|---|
| JUMP | JMP 303 | PCHL 351 | JNZ 302 | JZ 312 | JNC 322 | JC 332 | JPO 342 | JPE 352 | JP 362 | JM 372 |
| CALL (PUSH PC; JUMP) | CALL 315 | —— | CNZ 304 | CZ 314 | CNC 324 | CC 334 | CPO 344 | CPE 354 | CP 364 | CM 374 |
| RETURN (POP PC) | RET 311 | —— | RNZ 300 | RZ 310 | RNC 320 | RC 330 | RPO 340 | RPE 350 | RP 360 | RM 370 |

| RESTART | RST 0 307 | RST 1 317 | RST 2 327 | RST 3 337 | RST 4 347 | RST 5 357 | RST 6 367 | RST 7 377 |
|---|---|---|---|---|---|---|---|---|

## BIT MANIPULATION

| | | A | B | C | D | E | H | L | M |
|---|---|---|---|---|---|---|---|---|---|
| INCREMENT REGISTER | INR r | 074 | 004 | 014 | 024 | 034 | 044 | 054 | 064 |
| DECREMENT REGISTER | DCR r | 075 | 005 | 015 | 025 | 035 | 045 | 055 | 065 |

| | B, C | D, E | H, L | SP |
|---|---|---|---|---|
| INCREMENT REGISTER PAIR | INX B 003 | INX D 023 | INX H 043 | INX SP 063 |
| DECREMENT REGISTER PAIR | DCX B 013 | DCX D 033 | DCX H 053 | DCX SP 073 |

| | | | |
|---|---|---|---|
| ROTATE A | LEFT | RLC | 007 |
| | RIGHT | RRC | 017 |
| | LEFT THRU CARRY | RAL | 027 |
| | RIGHT THRU CARRY | RAR | 037 |

| | | | | |
|---|---|---|---|---|
| DECIMAL ADJUST ACCUMULATOR | DAA 047 | | | |
| COMPLEMENT A REGISTER | CMA 057 | | | |
| SET / COMPLEMENT CARRY | SET | STC 067 | COMP. | CMC 077 |

## ALU INSTRUCTIONS

| | | A | B | C | D | E | H | L | M | IMMED. | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD r TO A → A | ADD r | 207 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | ADI | 306 |
| ADD r + C TO A → A | ADC r | 217 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | ACI | 316 |
| SUB r FROM A → A | SUB r | 227 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | SUI | 326 |
| SUB (r + C) FROM A → A | SBB r | 237 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | SBI | 336 |
| AND r WITH A → A | ANA r | 247 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | ANI | 346 |
| EXCL. OR r WITH A → A | XRA r | 257 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | XRI | 356 |
| OR r WITH A → A | ORA r | 267 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | ORI | 366 |
| COMPARE r WITH A | CMP r | 277 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | CPI | 376 |

| | B, C | D, E | H, L | SP |
|---|---|---|---|---|
| DOUBLE ADD $r_2$, $r_1$ TO H, L → H, L | DAD B 011 | DAD D 031 | DAD H 051 | DAD SP 071 |

## MACHINE CONTROL

| | | |
|---|---|---|
| NO OPERATION | NOP | 000 |
| HALT | HLT | 166 |
| DISABLE INTERRUPT | DI | 363 |
| ENABLE INTERRUPT | EI | 373 |

## FLAGS

| | C Z S P |
|---|---|
| ALU INST | ✓ ✓ ✓ ✓ |
| DOUBLE ADD | ✓ — — — |
| DEC. ADJUST | ✓ ✓ ✓ ✓ |
| ROTATE | ✓ — — — |
| INCR./DECR. | – ✓ ✓ ✓ |
| INX / DCX | – – – – |
| SET/COMP. CARRY | ✓ – – – |
| POP PSW | ✓ ✓ ✓ ✓ |

REV. A

The *DEBUG version* of the *MIKE 3* microcomputer offers powerful tools for debugging hardware and software problems. It is also useful in development of 8080-based computers, and for educational purposes.

This set of operating instructions supplements the instructions provided for the basic *MIKE 3* microcomputer, OP811, and should be filed in the *MIKE 3 MANUAL* immediately following those instructions, in the OPERATING section of the manual.

---

*Note: for proper operation, the DEBUG system requires the following:*
*Model 471 CPU board, with 8080A;*
*Model 420 Console board (keyboard and LED digits);*
*Model 423 PROM/RAM board, with MIKE 3 Monitor, octal version, PROM 2200-17, D rev. or later, plus DEBUG PROM, 2200-24, C rev. or later; and*
*Model 472 DEBUG board.*
*The Model 472 DEBUG board also supports a 32-channel oscilloscope display circuit, for displaying the signals on all important microcomputer bus lines on an external triggered-sweep scope. For details, see the 472 data sheet.*

---

## SEC. 1 INTRODUCTION

The DEBUG system allows the user to step through programs one instruction at a time, using the Console keyboard. Unlike the step/run switch on other computers however--which typically deactivates the CPU between instructions--the DEBUG system maintains full processor capabilities between instruction steps.

In the single-step mode, the computer executes one instruction at a time, starting at any desired location, under control of the keyboard. After an instruction has been executed, the user can touch a key on the Console board and display the contents of the CPU registers; the 8080 flags; and memory, including the general purpose stack.

The single-step mode is useful in tracing programs stored in either PROM or RAM memory. Between steps, the contents of any register (or the flags or RAM memory) can be changed by the user.

The user can also set a *trap* at any desired location in memory. The computer can then be started at any other desired point in its program, and continues executing instructions until the trap point is met. Program execution now halts, and the user can display the contents of CPU registers, flags, etc., in the same manner as afforded by the single-step mode.

*SEC. 1 INTRODUCTION (cont'd)*

The DEBUG system relies on a special-purpose interrupt circuit (on the 472 DEBUG board) which requests an interrupt from the CPU two instructions after a designated output instruction has been executed. The system also makes heavy use of the general-purpose stack, a portion of RAM memory which (in the *MIKE 3*) extends downwards starting at location 013 377.

*SEC. 2 THE MONITOR MODE--OR "CCC MODE"*

All of the normal programming functions of the basic *MIKE 3* Monitor remain in effect in the DEBUG system--until the 9 key is pressed. This key will cause the computer to start executing the program beginning at location 001 000--which is the starting address of the DEBUG PROM.

The Operating Instructions for the basic *MIKE 3* system, OP811, should be consulted for instructions on how to program when in the CCC mode.

*SEC. 3 SINGLE-STEP MODE--OR "9'S MODE"*

When the user desires to single-step through a program it is necessary to carry out the following procedure:

STEP 1) If the program to be debugged will reside in RAM memory, first enter the program instructions into RAM using the normal keyboard procedures for the *MIKE 3*, while the computer is still in the Monitor (CCC) mode.

STEP 2) Set up the H and L memory addresses to the starting address of the program being debugged, using the × and ÷ keys, while still in the CCC mode. As usual, the display will blink:

HHHLLL - CCCxyz - HHHLLL - CCCxyz, etc.,

where HHHLLL is the memory address; CCC shows that the computer is in the ordinary Monitor mode; and xyz is the first instruction code in the program to be debugged.

STEP 3) Press the 9 key on the keyboard. The display will now blink:

HHHLLL - 999xyz - HHHLLL - 999xyz, etc.,

that is, just as it did before, except that the nines indicate

modular micros martin research

*SEC. 3 SINGLE-STEP MODE--OR "9'S MODE" (cont'd)* REV. A

STEP 3) (cont'd) that the system is in the single-step mode.

STEP 4) Press the -N key on the keyboard, to make sure that the interrupt request line is free to accept interrupt requests from the 472 DEBUG board.

STEP 5) Press M1--that is, hold the M shift key down while tapping the 1 numeric key. This has no effect on the blinking display, but causes CPU status information to be saved in RAM.

STEP 6) Press the - key. The computer now executes the first instruction in the program under test. Then the display returns to a blinking condition--still in the 9's mode--at the next instruction in the program, alternating between the address of that next instruction, and its opcode. Note that this next instruction will not actually be executed by the computer until the - key is pressed again.

STEP 7) Examine the contents of the CPU's registers and flags, as they have been affected by the instruction just executed. Use the numeric keys 0 through 8 to display the registers, flags, and memory (as pointed to by the H,L pair), and the function keys × and ÷ to see the current program counter addresses--PCH and PCL respectively.

See the chart on page 5 for these 9's mode keyboard functions.

STEP 8) To leave the 9's mode and return to the Monitor, just press the 9 key once more.

*SEC. 4 A DEBUG PROGRAM EXAMPLE*

In order to develop further details on the 9's mode, this section presents a concrete program example, giving the user a better feeling of how the DEBUG system works. The following steps are the same as the eight presented in the preceding section--but with more detail.

STEP 1) First enter a test program into page 010 of RAM. Set up the H register to 010 and the L to 000, and enter the program shown on page 4. Basically, this program loads distinct arbitrary values into the CPU's internal registers (B, C, D, E, H, L, and A) and into memory (as referenced by the H,L register pair). Then it clears the A register. Next, these are stored on the microcomputer's general-purpose stack, through a series of PUSH instructions. Then the data is restored to the CPU through a series

MR

# MICROCOMPUTER CODING FORM

**PROGRAM:** SAMPLE PROGRAM — DEBUG SYSTEM **DATE:** 4/16/76

**BY:** DPM **PAGE** 4 **OF** DS472





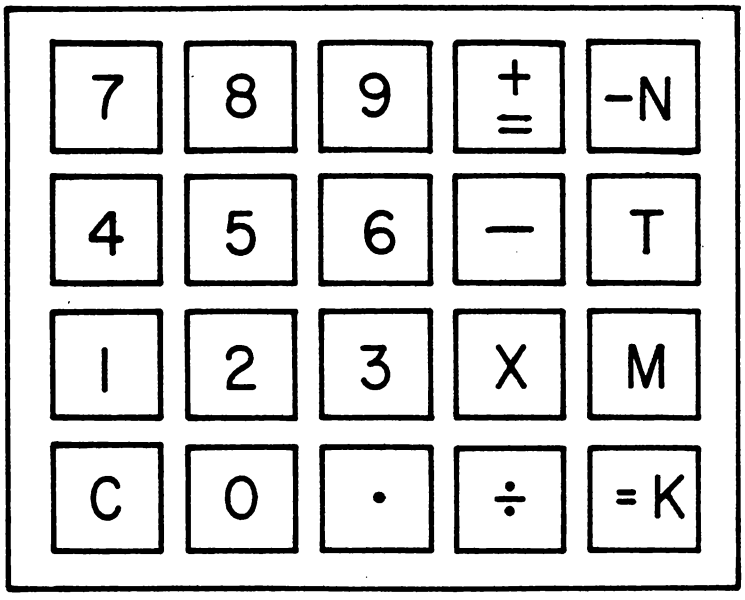
| | LOCATION H | LOCATION L | INSTRUCTION CODE 1 | INSTRUCTION CODE 2 | INSTRUCTION CODE 3 | LABEL | OPERATION | OPERANDS AND REMARKS |
|---|---|---|---|---|---|---|---|---|
| 1 | 010 | 000 | 006 | 007 | | TEST | MVI | B, 007 LOAD B REG. WITH 007 |
| 2 | | 002 | 016 | 010 | | | MVI | C, 010 C 010 |
| 3 | | 004 | 026 | 011 | | | MVI | D, 011 D 011 |
| 4 | | 006 | 036 | 012 | | | MVI | E, 012 E 012 |
| 5 | | 010 | 046 | 013 | | | MVI | H, 013 H 013 |
| 6 | | 012 | 056 | 014 | | | MVI | L, 014 L 014 |
| 7 | | 014 | 066 | 015 | | | MVI | M 015 MEMORY 015 |
| 8 | | 016 | 076 | 016 | | | MVI | A, 016 A REG. 016 |
| 9 | | 020 | 257 | | | | XRA | A CLEAR A REG. |
| 10 | | 021 | 365 | | | | PUSH | PSW SAVE A, FLAGS ON STACK |
| 11 | | 022 | 305 | | | | PUSH | B, SAVE B AND C |
| 12 | | 023 | 325 | | | | PUSH | D, SAVE D AND E |
| 13 | | 024 | 345 | | | | PUSH | H, SAVE H AND L |
| 14 | | 025 | 365 | | | | PUSH | PSW SAVE A, FLAGS AGAIN |
| 15 | | 026 | 063 | | | | INX | SP DISPLACE BY ONE |
| 16 | | 027 | 361 | | | | POP | PSW RESTORE A, FLAGS |
| 17 | | 030 | 341 | | | | POP | H, L |
| 18 | | 031 | 321 | | | | POP | D, E |
| 19 | | 032 | 301 | | | | POP | B, C |
| 20 | | 033 | 063 | | | | INX | SP DISPLACE BY ONE MORE |
| 21 | | 034 | 303 | 000 | 010 | | JMP | 010000 START AGAIN |
| 22 | | | | | | | | |
| 23 | | | | | | | | |
| 24 | | | | | | | | |
| 25 | | | | | | | | |
| 26 | | | | | | | | |



REORDER IN PADS OF 50 - CAT. NO. SP7080 - $2.00 - MARTIN RESEARCH, NORTHBROOK, IL 60062

*SEC. 4 A DEBUG PROGRAM EXAMPLE (cont'd)*

9's Mode Keyboard Function Chart




## KEYBOARD FUNCTIONS

| | *NO SHIFT KEYS* |
|---|---|
| KEY | HEX FUNCTION |
| 0 | *00* DISPLAY B REG. |
| 1 | *01* DISPLAY C REG. |
| 2 | *02* DISPLAY D REG. |
| 3 | *03* DISPLAY E REG. |
| 4 | *04* DISPLAY H REG. |
| 5 | *05* DISPLAY L REG. |
| 6 | *06* DISPLAY MEMORY (H,L) |
| 7 | *07* DISPLAY A REG. |
| 8 | *08* DISPLAY PSW (FLAGS) |
| 9 | *09* RETURN TO MONITOR |
| ± | *0A* SET TRAP POINTER |
| - | *0b* STEP TO NEXT INST. (OR TRAP) |
| × | *0C* DISPLAY PCH |
| ÷ | *0d* DISPLAY PCL |
| · | *0E* ENTER/EXIT TRAP MODE |
| C | *0F* CONTINUE (UP ONE FLOOR) |

| | M *KEY PRESSED* | =K *KEY PRESSED* | M *AND* =K *KEYS PRESSED* |
|---|---|---|---|
| KEY | HEX FUNCTION | HEX FUNCTION | HEX FUNCTION |
| 0 | *10* SET H,L=PC (STACK) | *20* | *30* |
| 1 | *11* DOWN ONE FLOOR | *21* | *31* |
| 2 | *12* INC. TRAP POINTER | *22* | *32* |
| 3 | *13* GO TO PROM 2 (002 000) | *23* | *33* |
| 4 | *14* | *24* | *34* |
| 5 | *15* | *25* | *35* |
| 6 | *16* | *26* | *36* |
| 7 | *17* | *27* | *37* |
| 8 | *18* | *28* | *38* |
| 9 | *19* | *29* | *39* |
| ± | *1A* | *2A* | *3A* |
| - | *1b* | *2b* | *3b* |
| × | *1C* | *2C* | *3C* |
| ÷ | *1d* | *2d* | *3d* |
| · | *1E* | *2E* | *3E* |
| C | *1F* | *2F* | *3F* |

| | |
|---|---|
| -N | *PRESET INTERRUPT* |
| T | *INTERRUPT* |

*Note: suggested practice--press in this order: T; -N.*

*SEC. 4 A DEBUG PROGRAM EXAMPLE (cont'd)* REV. A

STEP 1) (cont'd) of POP instructions. And, the program loops back again and repeats.

Use the normal write procedures for entering a program into memory, as shown in the basic *MIKE 3* Operating Instructions, OP811. When finished, go back to page 010, location 000, and read through the program, using the - key, in the Monitor mode, to be sure that the correct values have been stored in RAM.

STEP 2) Return the H,L address to 010 000 (with the ÷ key).

STEP 3) Press the 9 key on the keyboard. The display blinks:

999006 - 010000 - 999006 - 010000, etc.,

where 999 shows that the computer is in the 9's mode; 006 is the opcode for the first byte of the first instruction in the program on page 4; and 010 000 is the memory address.

STEP 4) Press the -N key on the keyboard, to make sure that the interrupt line is enabled. In general it is good practice, when using the T and -N keys to interrupt the *MIKE 3*, to do so in the following order: T, -N. This ensures that -N is always the last key pressed.

*Continuing earlier practice, these italicized paragraphs give technical details that the user may wish to skip over until he familiarizes himself with the operations of the DEBUG system.*

*The DEBUG system relies on the CPU's interrupt feature. The Model 472 DEBUG board includes a circuit which requests an interrupt from the CPU shortly after a designated instruction is executed by the DEBUG program, which is stored in the DEBUG PROM. The software in the DEBUG PROM distinguishes between interrupts requested in this manner, and an overriding interrupt generated by the user when he presses the T key on the Console keyboard. However, when the T key is used, the interrupt request line, $\overline{INTR}$, is held low by a flip-flop on the 420 Console board, preventing the 472 board from requesting interrupts for the purposes of the DEBUG system. Thus, the -N key should always be pressed following the use of the T key, in order to enable the DEBUG system.*

STEP 5) Press M1 (hold down the M shift key and tap the 1 numeric key). The display continues blinking, in the 9's mode. CPU status information has been saved in RAM.

*SEC. 4 A DEBUG PROGRAM EXAMPLE (cont'd)*

STEP 5) (cont'd) *When the M1 key combination occurs while the computer is in the 9's mode, the DEBUG software interprets this as a request to save CPU status in the microcomputer's general-purpose stack. A series of PUSH instructions are executed, storing the contents of CPU registers, flags, and program counter in RAM memory, starting at location 013 377 and moving downwards. Taken together, the ten memory words which represent the stored CPU status are termed a* floor *in the stack, and thus the M1 key is labeled, on page 5,* DOWN ONE FLOOR. *Because CPU information is stored in the stack, external to the CPU, CPU operations are available between each instruction step for inspection and modification by the user. This principle is the key to the power of the DEBUG system.*

STEP 6) Press the - key. The computer executes the first instruction in our test program, namely, MVI B, 007. That is, the 8080's internal B register has been loaded with the octal value 007. Now the display begins to blink again:

999016 - 010002 - 999016 - 010002, etc.,

where 999 indicates that the computer is still in the 9's mode; 016 is the first byte of the following instruction; and 010 002 is the location of the first byte of that next instruction.

Note that the display has jumped from location 010 000 to 010 002, apparently skipping 010 001. Looking at the program listing on page 4, notice that the data at location 001 is merely the second byte of the first instruction.

The instruction opcode 016 appears in order to allow the user to examine the next instruction. However, this next instruction will not actually be executed by the computer until the - key is pressed once more.

*When the computer is in the 9's mode and the - key is pressed, causing the computer to step through the next instruction, the following events occur. First, the stack pointer moves up one floor--that is, the saved CPU status information is popped into the 8080. Then the 8080 executes the next instruction. Finally the stack pointer moves down one floor--in other words, the new CPU status information, as it has been affected by the instruction just executed, is stored once more on the stack.*

REV. A

*SEC. 4 A DEBUG PROGRAM EXAMPLE (cont'd)*

STEP 7) Examine the contents of the CPU's registers and flags. Referring to the chart on page 5, this examination proceeds as follows:

KEY 0-- DISPLAY B REGISTER. Since the first instruction just loaded the B register with 007, this is the value that appears on the display. The user familiar with 8080 instruction codes will notice that 0 is the code for the B register in the 8080's microcoding, making these key assignments easy to learn and remember.

KEY 1-- DISPLAY C REGISTER. This register has not yet been loaded up by our sample program, and still contains some arbitrary value--typically 377 (left over from other operations in the DEBUG and MONITOR programs). If this key is pressed after the sample program has progressed further, the value 010 would appear.

KEY 2-- DISPLAY D REGISTER. Again, the D register has not yet been loaded at this point by our sample program, so it typically has the left-over value 231.

KEY 3-- DISPLAY E REGISTER. Typical left-over value, 000.

KEY 4-- DISPLAY H REGISTER. The high-order memory address in the 8080's H register reads 010, the location in memory in which the sample program was loaded.

KEY 5-- DISPLAY L REGISTER. The low-order memory address is still 000, the value left over from when the sample program was loaded in.

KEY 6-- DISPLAY MEMORY. This is the contents of the memory word pointed to by the H and L register values (above) and reads 006--the first byte of the first instruction in the sample program.

KEY 7-- DISPLAY A REGISTER. A left-over value, typically 100.

KEY 8-- DISPLAY FLAGS. The flags are displayed in the form of a *processor status word*, or *PSW*. The chart on the next page shows how the 20 possible values of the PSW correspond to the 8080's flags.

*SEC. 4 A DEBUG PROGRAM EXAMPLE (cont'd)*

KEY 8-- DISPLAY FLAGS (cont'd)

| PSW | S | Z | AC | P | CY |
|---|---|---|---|---|---|
| 002 | | | | | |
| 003 | | | | | ✓ |
| 006 | | | | ✓ | |
| 007 | | | | ✓ | ✓ |
| 022 | | | ✓ | | |
| 023 | | | ✓ | | ✓ |
| 026 | | | ✓ | ✓ | |
| 027 | | | ✓ | ✓ | ✓ |
| 106 | | ✓ | | ✓ | |
| 107 | | ✓ | | ✓ | ✓ |
| 126 | | ✓ | ✓ | ✓ | |
| 127 | | ✓ | ✓ | ✓ | ✓ |
| 202 | ✓ | | | | |
| 203 | ✓ | | | | ✓ |
| 206 | ✓ | | | ✓ | |
| 207 | ✓ | | | ✓ | ✓ |
| 222 | ✓ | | ✓ | | |
| 223 | ✓ | | ✓ | | ✓ |
| 226 | ✓ | | ✓ | ✓ | |
| 227 | ✓ | | ✓ | ✓ | ✓ |

S = Sign
Z = Zero
AC = Auxiliary carry
P = Parity
CY = Carry

A check mark (✓) in the table to the left means that the designated flag is *set*--ie, logic high (a 1).

*The table to the left must be modified slightly for the extra SUB flag, when using the uPD 8080 from NEC.*

KEY *-- DISPLAY PCH. This is the high-order memory address in the 8080 program counter, that is, the location at which instructions are currently being executed. Reads 010.

KEY ÷-- DISPLAY PCL. The low-order program counter address. Reads 002. The PC has already stepped to page 010, location 002, in preparation for the execution of the following instruction.

STEP 8) Having just inspected the program counter address, press the 9 key. The computer leaves the 9's mode and returns to the Monitor. Note the resulting display:

CCC002 - 013362 - CCC002 - 013362, etc.,

where the CCC indicates that the computer is back in the Monitor mode; 002 is the contents of the PCL--and 013 362 is the memory address.

*Why is the memory location 013 362? Answer: this is the location on the computer's general-purpose stack at which the*

modular micros martin research

*SEC. 4 A DEBUG PROGRAM EXAMPLE (cont'd)*

STEP 8) (cont'd) *DEBUG program has stored the value of the PCL. Press the - key, and read the value at the next location in memory, 013 363, displaying the value of the PCH, namely, 010. This illustrates the principle that, when viewing the contents of CPU registers while in the DEBUG mode, one is actually displaying the values stored in the stack. Since the stack is effectively refreshed each and every time that an instruction is executed, when in the DEBUG mode, the CPU values seen in the stack correspond exactly to those within the 8080's internal registers.*

*Note that the DEBUG PROM is always using an area of the stack directly below the area being used by the program being executed. The DEBUG system can, for example, single-step through PUSH and POP instructions--which themselves use the stack--with no problems. The only restriction: there must always be at least one floor, or ten (decimal) stack locations, below the lowest stack address being used by the program being debugged, for the use of the DEBUG system. This is not a practical limitation, since nearly all of page 013 in RAM is available--all the way down to location 013 353, which (as explained below) is used for storage of special DEBUG program information.*

We have left the 9's mode for the purposes of illustration. But to return to the single-step mode again, start the above procedure again, beginning with Step (2). This time, at Step (6), press the - key twice--thus stepping through the first two instructions. The display will blink:

999026 - 010004 - 999026 - 010004, etc.,

where 026 is the first byte of the *third* instruction in the sample program, and 010 004 is the address of the first byte of that instruction. Examine CPU registers and flags; note that the C register has now been loaded up with 010, as called for by the program. Do not leave the 9's mode, but press the - key again. The computer will step to the next instruction, executing it, and begin blinking once again.

modular micros martin research

*SEC. 5 TRAP MODE--OR "b'S MODE"* REV. A

Using the DEBUG system, the user can set up a *trap* at any desired location. Starting program execution at any other desired location, the computer executes each instruction (in the DEBUG mode, saving status between instructions) until it hits the designated trap address. The trap address can reside at any location in memory, including permanently-stored PROM or ROM memory. Here is an example:

STEP A) Before entering either the single-step or trap modes, first set up a trap address while still in the CCC (Monitor) mode. Let it be 010 010, a location a few steps down through the sample program (Sec. 5 above). Enter this address using the × and ÷ keys, as usual for Monitor program address entry.

STEP B) Enter the 9's mode by pushing the 9 key. Display blinks:

999046 - 010010 - 999046 - 010010

where 046 is the instruction stored as part of the sample program at location 010 010.

STEP C) Push the ± key, while still in the 9's mode. This enters the trap address.

*The DEBUG program actually enters the trap address at a specially designated address in RAM, deep down in the general-purpose stack at locations 013 050 and 013 051. Location 013 052 contains an eight-bit flag which is used to indicate the source of an interrupt--i.e., Console or 472 DEBUG board. Location 013 053 contains another eight-bit flag which indicates when the system is in the trap mode.*

STEP D) Go back to the Monitor mode and set up the memory address where program execution is to begin. Let this be 010 000, the beginning of our sample program. Press 9; display blinks:

CCC046 - 010010 - CCC046 - 010010, etc.

Change the low-order address to 000 by pressing the ÷ key. Display blinks:

CCC006 - 010000 - CCC006 - 010000, etc.

STEP E) Prepare for debugging in the same manner as for the 9's mode. First, push 9. Push -9 to ensure interrupt enabling. Push M1 to store status information on the stack.

*SEC. 5 TRAP MODE--OR "b'S MODE" (Continued)* REV. A

STEP F) Push the · key to enter the trap mode. The display now blinks:

bbb046 - 010010 - bbb046 - 010010, etc.,

where the bbb indicates that the computer is in trap mode; 010 010 is the trap address; and 046 is the word in memory at the trap address (from our sample program). Note that at this point the actual program counter address is still 010 000, as set up by Step (D).

STEP G) Push the - key. Instead of stepping through only one instruction, the computer steps through all of the instructions it encounters, in program sequence, until it is about to execute the instruction at the trap address. Here it stops, immediately before the trap location, and blinks:

999046 - 010010 - 999046 - 010010, etc.

Note that when the system encounters the trap, it automatically reverts from the b's mode to the 9's mode.

During the last step, Step (G), all the program instructions starting at 010 000, through 010 007, were executed by the computer. The user can use the 9's mode to satisfy himself that the CPU registers have been loaded with the proper values.

As stated above, when the computer has entered the trap mode, the display blinks as follows:

bbbABC - HHHLLL - bbbABC - HHHLLL, etc.,

where HHH LLL is the address of the trap pointer, and ABC is the data word at that location in memory. If the user now presses the · key again, the computer returns to the 9's mode. Now the display blinks back and forth, but the memory address displayed is no longer the trap pointer location, but the current program counter location. In other words, the display indicates the next instruction to be executed in the DEBUG mode. This helps the user keep track of the starting location in the DEBUG process.

The DEBUG program causes the right-hand pair of digits on the display board to count rapidly during execution of programs, while in the trap mode. However, if the program sequence is very short--as it was in the above example--this counting will probably go unnoticed. The purpose of the counting display is to assure the user that program execution is taking place, during long program sequences.

The program listing on a following page has been chosen as an example of one which takes a few seconds for the computer to execute while in trap

*SEC. 5 TRAP MODE--OR "b'S MODE" (Continued)* REV. A

mode. It is typical of an 8080 program used to develop a time delay. Essentially, it loads the A register with 377; loads the B, C register pair with 005 000 (split octal), or 1,280 (decimal); and decrements the B register. This process repeats until the B register compares with the A--and when it does, the program falls through the conditional branch instruction. The trap pointer will be set to stop program execution at this point.

STEP W) In the Monitor mode, enter the program listing, starting at RAM location 010 000. Check program entry by using the - key while in the Monitor mode.

STEP X) While still in the Monitor mode, enter 010 012 as the memory address using the × and ÷ keys. Enter the 9's mode by pressing 9, and set the trap pointer at this location by pressing ±.

STEP Y) Return to the Monitor mode momentarily, and press the ÷ key, making the current memory address 010 000.

STEP Z) Return to 9's mode. Initialize by pressing the -N key and the M1 key combination. Enter trap mode by pressing the · key. Push the - key to start executing instructions. The right-hand two digits will count rapidly for about four seconds, then stop, blinking:

999000 - 010012 - 999000 - 010012, etc.

Note that this procedure involves the execution not only of over 1,000 iterations of the program loop, but also includes a full status-restoring routine before each instruction, and a status-saving routine after each instruction (just as in the single-step mode). Plus, each time that the computer executes 256 of the instructions in the program being debugged by the user (not counting DEBUG program instructions necessary for saving and restoring CPU status), it increments the values displayed on two right-hand digits. Yet, the entire operation takes only a few seconds.

MR

# MICROCOMPUTER CODING FORM

PROGRAM: TRAP MODE DEMONSTRATION

DATE: 4/76

BY: DPM

PAGE 14 OF OP814 REV. A

| | LOCATION H | LOCATION L | INSTRUCTION CODE 1 | INSTRUCTION CODE 2 | INSTRUCTION CODE 3 | LABEL | OPERATION | OPERANDS AND REMARKS |
|---|---|---|---|---|---|---|---|---|
| 1 | 010 | 000 | 076 | 377 | | | MVI | A, 377 LOAD A REGISTER |
| 2 | | 002 | 001 | 000 | 005 | | LXI | B, 005000 LOAD B, C PAIR |
| 3 | | 005 | 013 | | | LOOP | DCX | B DECREMENT B, C PAIR |
| 4 | | 006 | 270 | | | | CMP | B COMPARE B WITH A |
| 5 | | 007 | 302 | 005 | 010 | | JNZ | LOOP IF B≠A, REPEAT |
| 6 | | 012 | 000 | | | | NOP | DONE |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | | | | | | | | |
| 15 | | | | | | | | |
| 16 | | | | | | | | |
| 17 | | | | | | | | |
| 18 | | | | | | | | |
| 19 | | | | | | | | |
| 20 | | | | | | | | |
| 21 | | | | | | | | |
| 22 | | | | | | | | |
| 23 | | | | | | | | |
| 24 | | | | | | | | |
| 25 | | | | | | | | |
| 26 | | | | | | | | |



REORDER IN PADS OF 50 - CAT. NO. SP7080 - $2.00 - MARTIN RESEARCH, NORTHBROOK, IL 60062

# 8080 INSTRUCTION SET

## REGISTER TRANSFERS

MOV B, H = MOVE TO THE B REG. THE CONTENTS OF THE H REGISTER.

A = 7 C = 1 E = 3 L = 5
B = 0 D = 2 H = 4 M = 6

| | | DESTINATION A | B | C | D | E | H | L | M |
|---|---|---|---|---|---|---|---|---|---|
| SOURCE | A | 177 | 107 | 117 | 127 | 137 | 147 | 157 | 167 |
| | B | 170 | 100 | 110 | 120 | 130 | 140 | 150 | 160 |
| | C | 171 | 101 | 111 | 121 | 131 | 141 | 151 | 161 |
| | D | 172 | 102 | 112 | 122 | 132 | 142 | 152 | 162 |
| | E | 173 | 103 | 113 | 123 | 133 | 143 | 153 | 163 |
| | H | 174 | 104 | 114 | 124 | 134 | 144 | 154 | 164 |
| | L | 175 | 105 | 115 | 125 | 135 | 145 | 155 | 165 |
| | M | 176 | 106 | 116 | 126 | 136 | 146 | 156 | — |
| MOVE IMMEDIATE | MVI r | 076 | 006 | 016 | 026 | 036 | 046 | 056 | 066 |

## DOUBLE-PRECISION TRANSFERS

| | | DESTINATION B, C | D, E | H, L | SP | PC |
|---|---|---|---|---|---|---|
| SOURCE | D, E | — | — | XCHG 353 | — | — |
| | H, L | — | XCHG 353 | — | SPHL 371 | PCHL 351 |
| | IMMED. | LXI B 001 | LXI D 021 | LXI H 041 | LXI SP 061 | — |

## INDIRECT TRANSFERS

| BYTES | INSTR. | REG. | DATA | MEMORY ADDR. BY: |
|---|---|---|---|---|
| 1 | STAX B 002 | A REGISTER | → | B, C |
| | LDAX B 012 | | ← | |
| | STAX D 022 | | → | D, E |
| | LDAX D 032 | | ← | |
| | MOV M, r SEE ABOVE | INDEX REG. r | → | H, L |
| | MOV r, M SEE ABOVE | | ← | |
| | MVI M 066 | 2ND BYTE OF INST. | → | |
| 2 | PUSH B 305 | B, C | → | SP (Stack Pointer) PUSH DECR. SP BY 2 POP INCR. SP BY 2 |
| | POP B 301 | | ← | |
| | PUSH D 325 | D, E | → | |
| | POP D 321 | | ← | |
| | PUSH H 345 | H, L | → | |
| | POP H 341 | | ← | |
| | PUSH PSW 365 | A, PSW | → | |
| | POP PSW 361 | | ← | |
| | XTHL 343 | H, L | ↔ | SP |

## DIRECT LOAD & STORE

| BYTES | INSTR. | REG. | DATA | MEMORY ADDR. BY: |
|---|---|---|---|---|
| 1 | STA 062 | A | → | 2nd AND 3rd BYTES OF INSTR. |
| | LDA 072 | | ← | |
| 2 | SHLD 042 | H, L | → | |
| | LHLD 052 | | ← | |

## INPUT/OUTPUT

| INPUT | OUTPUT |
|---|---|
| IN 333 | OUT 323 |
| 2ND BYTE = I/O ADDR. | |

| UNUSED | |
|---|---|
| 010 | 313 |
| 020 | 331 |
| 030 | 335 |
| 040 | 355 |
| 050 | 375 |
| 060 | — |
| 070 | — |

Copyright © 1976
MARTIN RESEARCH
ALL RIGHTS RESERVED

## PROGRAM CONTROL

| | UNCONDITIONAL | INDIRECT | ZERO FALSE | ZERO TRUE | CARRY FALSE | CARRY TRUE | PARITY FALSE | PARITY TRUE | SIGN FALSE | SIGN TRUE |
|---|---|---|---|---|---|---|---|---|---|---|
| JUMP | JMP 303 | PCHL 351 | JNZ 302 | JZ 312 | JNC 322 | JC 332 | JPO 342 | JPE 352 | JP 362 | JM 372 |
| CALL (PUSH PC; JUMP) | CALL 315 | — | CNZ 304 | CZ 314 | CNC 324 | CC 334 | CPO 344 | CPE 354 | CP 364 | CM 374 |
| RETURN (POP PC) | RET 311 | — | RNZ 300 | RZ 310 | RNC 320 | RC 330 | RPO 340 | RPE 350 | RP 360 | RM 370 |

| RESTART | RST 0 307 | RST 1 317 | RST 2 327 | RST 3 337 | RST 4 347 | RST 5 357 | RST 6 367 | RST 7 377 |
|---|---|---|---|---|---|---|---|---|

## BIT MANIPULATION

| | | A | B | C | D | E | H | L | M |
|---|---|---|---|---|---|---|---|---|---|
| INCREMENT REGISTER | INR r | 074 | 004 | 014 | 024 | 034 | 044 | 054 | 064 |
| DECREMENT REGISTER | DCR r | 075 | 005 | 015 | 025 | 035 | 045 | 055 | 065 |

| | B, C | D, E | H, L | SP |
|---|---|---|---|---|
| INCREMENT REGISTER PAIR | INX B 003 | INX D 023 | INX H 043 | INX SP 063 |
| DECREMENT REGISTER PAIR | DCX B 013 | DCX D 033 | DCX H 053 | DCX SP 073 |

| | | | |
|---|---|---|---|
| ROTATE A | LEFT | RLC | 007 |
| | RIGHT | RRC | 017 |
| | LEFT THRU CARRY | RAL | 027 |
| | RIGHT THRU CARRY | RAR | 037 |

| | | | | |
|---|---|---|---|---|
| DECIMAL ADJUST ACCUMULATOR | DAA 047 | | | |
| COMPLEMENT A REGISTER | CMA 057 | | | |
| SET / COMPLEMENT CARRY | SET | STC 067 | COMP. | CMC 077 |

## ALU INSTRUCTIONS

| | | A | B | C | D | E | H | L | M | IMMED. | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD r TO A → A | ADD r | 207 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | ADI | 306 |
| ADD r + C TO A → A | ADC r | 217 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | ACI | 316 |
| SUB r FROM A → A | SUB r | 227 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | SUI | 326 |
| SUB (r + C) FROM A → A | SBB r | 237 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | SBI | 336 |
| AND r WITH A → A | ANA r | 247 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | ANI | 346 |
| EXCL. OR r WITH A → A | XRA r | 257 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | XRI | 356 |
| OR r WITH A → A | ORA r | 267 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | ORI | 366 |
| COMPARE r WITH A | CMP r | 277 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | CPI | 376 |

| | B, C | D, E | H, L | SP |
|---|---|---|---|---|
| DOUBLE ADD $r_2$, $r_1$ TO H, L → H, L | DAD B 011 | DAD D 031 | DAD H 051 | DAD SP 071 |

## MACHINE CONTROL

| | | |
|---|---|---|
| NO OPERATION | NOP | 000 |
| HALT | HLT | 166 |
| DISABLE INTERRUPT | DI | 363 |
| ENABLE INTERRUPT | EI | 373 |

## FLAGS

| | C Z S P |
|---|---|
| ALU INST | ✓ ✓ ✓ ✓ |
| DOUBLE ADD | ✓ — — — |
| DEC. ADJUST | ✓ ✓ ✓ ✓ |
| ROTATE | ✓ — — — |
| INCR./DECR. | — ✓ ✓ ✓ |
| INX / DCX | — — — — |
| SET/COMP. CARRY | ✓ — — — |
| POP PSW | ✓ ✓ ✓ ✓ |

# 6. The Central Processing Unit (CPU)

The *471 CPU* is a complete 8-bit central processor on a single printed circuit board. Based on the 8080A microprocessor, the 471 generates the essential control signals for a complete *MIKE 3* microcomputer system.

The 471 supports the full capabilities of the 8080A microprocessor, including high speed--2.0 us instruction cycle--and memory addressing of up to 64K (65,536) bytes. The instruction set supports logical (Boolean), arithmetic, and bit-manipulating operations; the CPU can convert data to decimal (BCD) notation with a single instruction. Logical operations set flags which may be tested by conditional branch instructions to control program flow. For added convenience in memory addressing, six general-purpose internal registers may be addressed not only singly, but in pairs. There are also double-precision (16-bit) addition and increment/decrement instructions.

The 8080 stack pointer addresses external RAM memory, allowing for practically unlimited nesting of program subroutines. Through single-byte PUSH and POP instructions, registers, flags, and the program counter can be saved in the external stack, and restored later--greatly facilitating interrupt handling. The 471 includes circuitry for three interrupt levels, expandable to an eight-level prioritized interrupt system by adding an accessory circuit board. A digital timer restarts the computer automatically should the CPU fail to recognize interrupt requests for a set period.

Power bus drivers are included for driving expanded systems. However, control by the 8080A of the data and address busses may be suspended by a HOLD request, thereby allowing for direct memory access (DMA) or multi-processor operation.

The 471 includes a reliable crystal-controlled clock. Specialized control signals are developed on the CPU board and are present on the system bus to simplify expansion, by minimizing the extra interface circuitry needed on each new memory or input/output board. These signals not only include those required for 8080-type I/O instructions, but also support the 6800/6502 and 8008 I/O addressing modes. The 50-pin bus structure is compatible with these and other eight-bit processors, allowing the 471 to be used interchangeably with other CPU boards in interfacing to memory and accessory modules in the *modular micro* series.

While supporting all the potential of the powerful 8080A microprocessor, the Model 471 CPU board is a good example of the flexibility of the *modular micro* approach. Optimal for small systems--yet fully expandable.

## APPLICATIONS

*Industrial* The flexible *modular micro* approach makes the Model 471 ideal for prototyping computers based on the 8080A microprocessor. The 471 is also attractive to the OEM for pilot runs and low-volume production, and can easily be adapted for specialized purposes. Ideal for a whole new generation of intelligent machines, including industrial controllers, computer peripherals, terminals, business machines, and many others.

*Educational* In design and packaging, the *modular micro* concept is well-suited for educational purposes. Used by university engineering and computer science departments in formal course work, and by engineers and programmers to keep up with the state of the art.

# TECHNICAL DATA

## SEC. 1 CIRCUIT DESCRIPTION

1.1 *Introduction* The 471 CPU board is based on the 8080 microprocessor. Normally the most complex element in a *MIKE 3* microcomputer, the CPU generates nearly all of the control signals which appear on the system's bus structure.

To form a working microcomputer, the 471 CPU board must be interfaced with memory storage. For example, the Model 423 PROM/RAM board includes space for PROM memory from which to read permanently-stored program instructions (such as the *MIKE 3* Monitor), and RAM memory for temporary data storage and for the push-pop stack. In addition, a complete microcomputer usually includes input/output boards for interfacing with keyboards, displays, printers, and other peripheral devices.

This circuit description conforms to the breakdown of the schematic diagram (beginning on page 4 of this data sheet), as follows:

| | | | |
|---|---|---|---|
| 1.2 | 8080 CPU | 1.8 | Interrupt Logic |
| 1.3 | Bus Drivers | 1.9 | Hold Circuitry |
| 1.4 | Main Timing | 1.10 | Wait Circuitry |
| 1.5 | System Control Logic | 1.11 | Indicator Lights |
| 1.6 | I/O Instruction Modes | 1.12 | Power Requirements |
| 1.7 | Reset Circuitry | | |

1.2 *8080 CPU* The 8080 is a microprocessor within a single integrated circuit. Based on 17-volt N-channel MOS technology, its 40-pin package includes terminals for an eight-bit bidirectional data bus, D7-DØ; a sixteen-bit address bus, A15-AØ; power supplies of +12 V, +5 V, and -5 V; and a variety of control signals. The 8080 requires an externally-generated two-phase clock (PH1 and PH2) driven to +12 V; all other control signals are at TTL voltage levels, between 0 and +5 volts.

Complete functional and electrical characteristics of the 8080 are provided in the manufacturers' manuals and spec sheets. Essential details are reprinted by Martin Research in data sheet DS2204. The reader should refer to these sources for an explanation of internal processor operations, and for detailed information on the 8080 instruction set. This data sheet focuses on the circuitry of the Model 471, a CPU board in the *modular micro* series from Martin Research.

*NOTICE CONCERNING 8080 A VERSION* - - The 8080A is an improved version of the 8080, but is compatible with early 8080 devices. Changes implemented with the introduction of the 8080A include the following: a relaxation of the minimum clock input high voltage from +11.0 to +9.0 V; improved current sinking capabilities on output terminals, from 1.7 ma on the data bus (0.75 mA at other outputs) to 1.9 mA on all output terminals, measured at 0.45 V; and internal synchronization of the HOLD signal, avoiding the need for synchronizing circuitry outside the 8080.

SEC. 1.2 *8080 CPU* (cont'd.)

The 471 CPU board has been designed for use with the Intel C8080A, Advanced Micro Devices Am9080A, or NEC Microcomputers uPD8080A. The Texas Instruments TMS 8080, and early Intel C8080, will work with the 471 CPU board; however, please note the following. The 8224 clock driver circuit is not guaranteed to deliver PH1 and PH2 clock voltages of greater than +9.4 V, while the TMS 8080 and C8080 are specified to require +11.0 V. Lab tests by Martin Research indicate that in practice these components do interface successfully. Nevertheless, we cannot interpose a guarantee to that effect where the manufacturer's specifications fail to do so.

The NEC Microcomputers uPD8080A includes two features not found on other versions of the 8080: first, the DAA (decimal adjust accumulator) instruction successfully converts binary data to BCD after a subtraction instruction (as well as addition). Second, a MOV r,r (register-to-register transfer) instruction takes four (rather than five) machine cycles to execute. This last feature improves uPD8080A speed, but means that software is not 100% compatible with other 8080 processors. That is, slight differences in programming will be necessary in programmed timing loops which depend for accuracy on an exact count of machine cycles executed.

1.3 *Bus Drivers* Like other MOS microprocessors, the internal circuitry of the 8080A does not have the power to drive the large array of memory and other devices typically connected to the *modular micro* bus. For this reason, the 471 includes bus drivers both on the data bus and on the address bus.

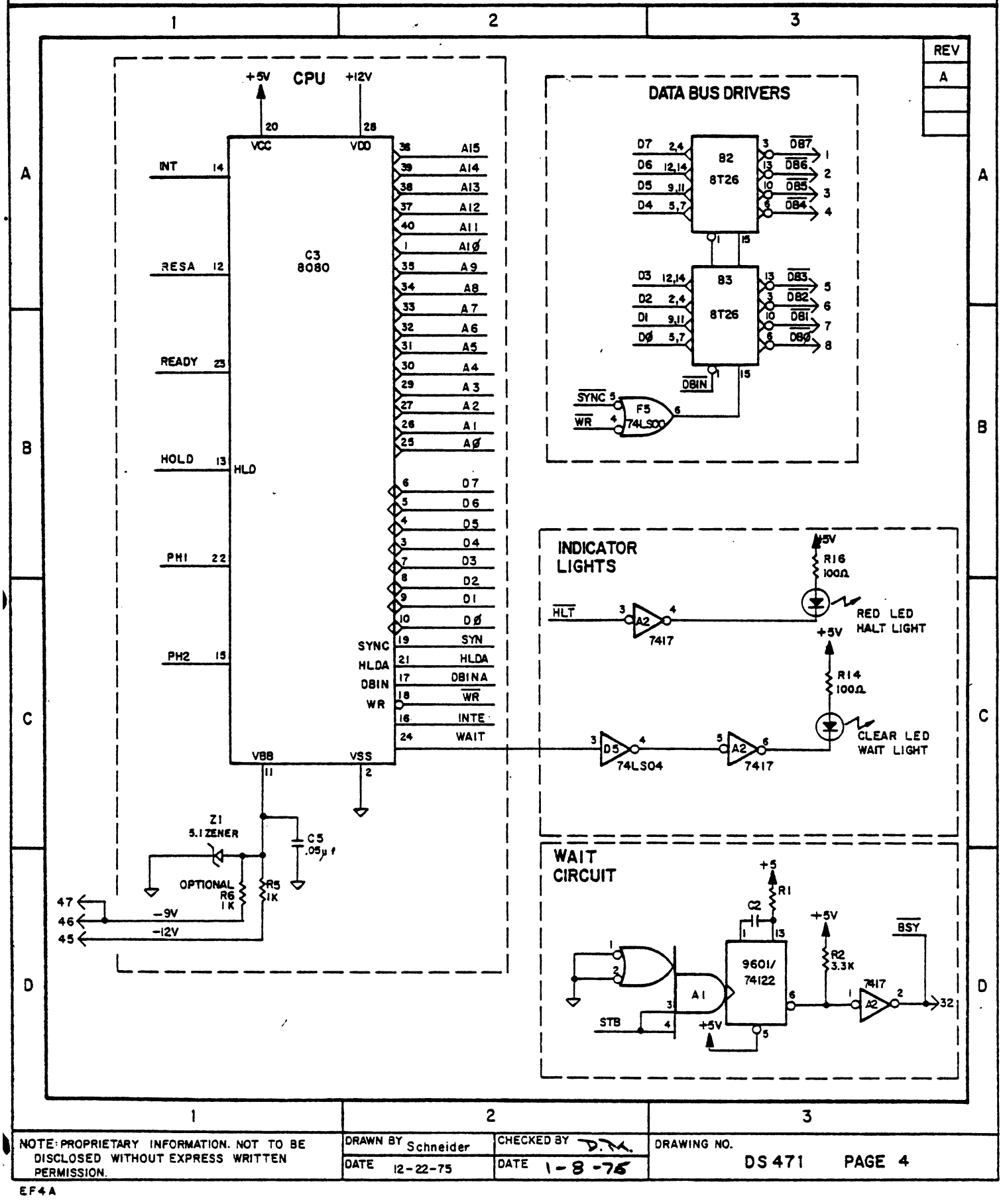
The data bus driver stages include two type 8T26 inverting bus drivers. Data traveling on the *modular micro* data bus is inverted in computers using the 471 CPU board. Electrically, this improves system noise immunity. Accessory memory boards using PROM memory normally also use inverting bus drivers, so inversion of data when programming PROMs is unnecessary. In the case of RAM, there is no problem even when non-inverting bus drivers are used on the RAM board: if data is inverted when written into RAM, it is reinverted (and restored) when read back. Input and output data, however, is affected.

The DBIN signal, produced by the 8080A, indicates that the CPU is ready to read data from memory or an input device. Inverted by the system control logic (Sec. 1.5), $\overline{\text{DBIN}}$ activates the inward-pointing drivers of the two 8T26 ICs. The outward-pointing drivers are activated by SYNC and $\overline{\text{WR}}$, both CPU-produced signals--the first indicating that machine status information is present on the bus. When neither pin 1 nor 15 is activated, the data termianls of the 8T26s are in a high-impedance state, and do not load the data bus.

The Address/Control Bus Driver stage includes four type 8T97 hex tri-state bus drivers. All 24 driver circuits are normally active, driving the 16-line *modular micro* address bus, as well as eight selected control lines. Only when the CPU has acknowledged a HOLD request are these drivers deactivated, allowing some device other than the 471 CPU board to control the system. Useful in DMA and multi-processor applications, this provision is discussed further in Sec. 1.9.

DS 471
PAGE 4

# CENTRAL PROCESSING UNIT
## MODEL 471



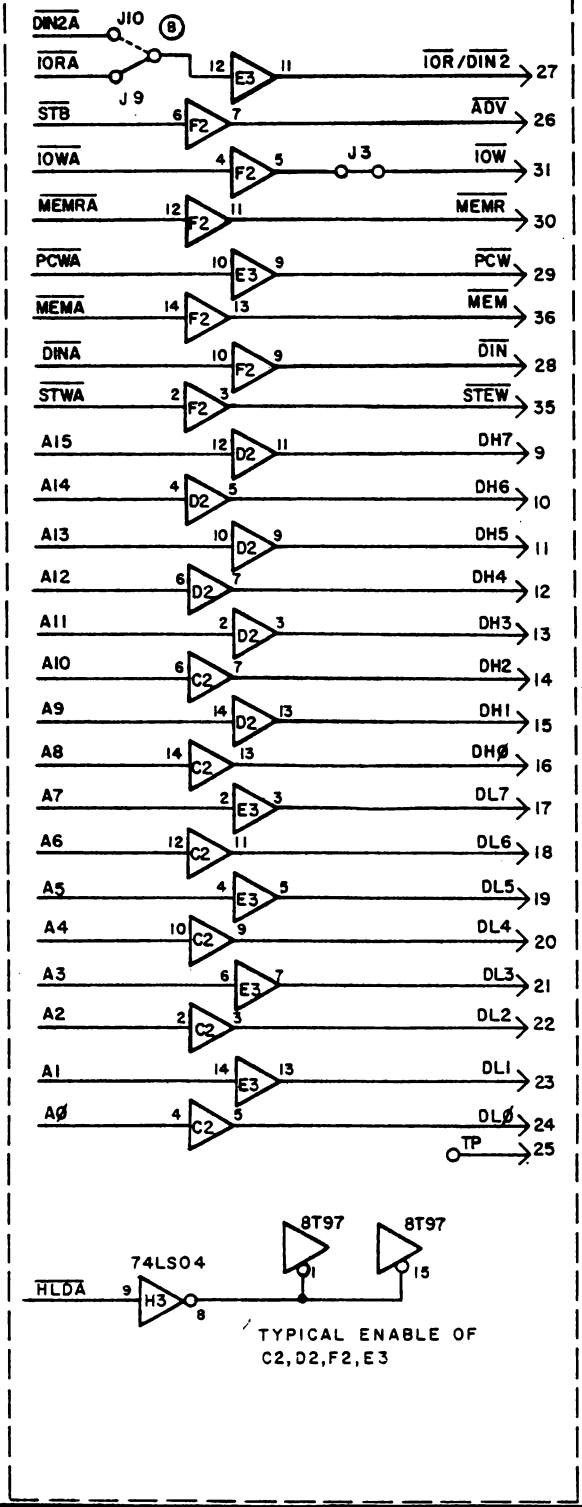

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY D.M. | DRAWING NO. |
|---|---|---|---|
| | DATE 12-22-75 | DATE 1-8-76 | DS 471 PAGE 4 |

EF4A

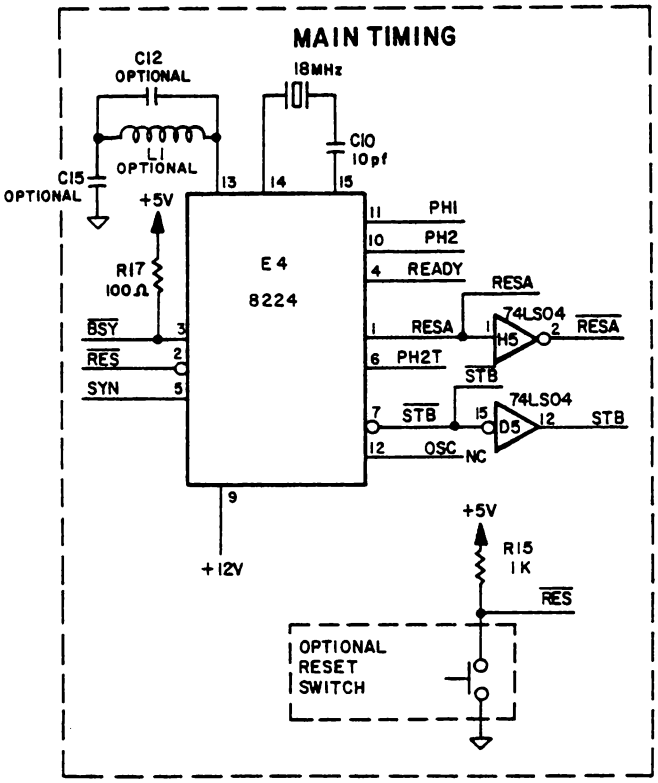
modular micros martin research

| REV |
|---|
| A |
| B |
| |

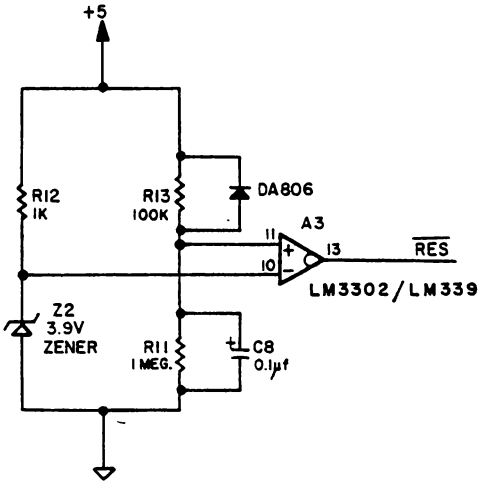
## ADDRESS/CONTROL BUS DRIVERS




## MAIN TIMING




## POWER ON RESET



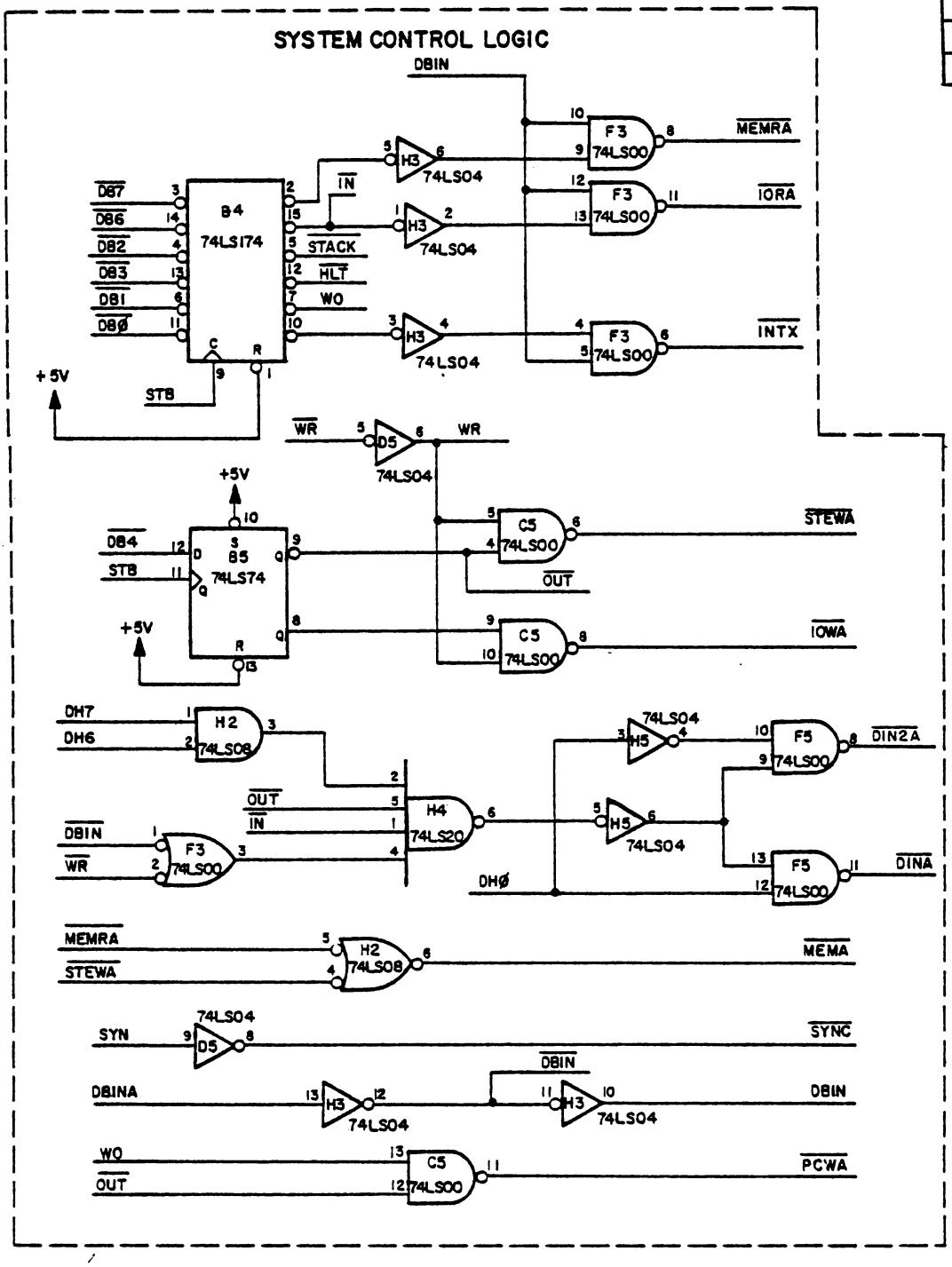

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY D.M. | DRAWING NO. |
|---|---|---|---|
| | DATE 12-22-75 | DATE 1-8-76 | DS 471 PAGE 5 |

EF4A

modular micros | martin research

# CENTRAL PROCESSING UNIT
## MODEL 471

| REV |
|---|
| A |
| |
| |



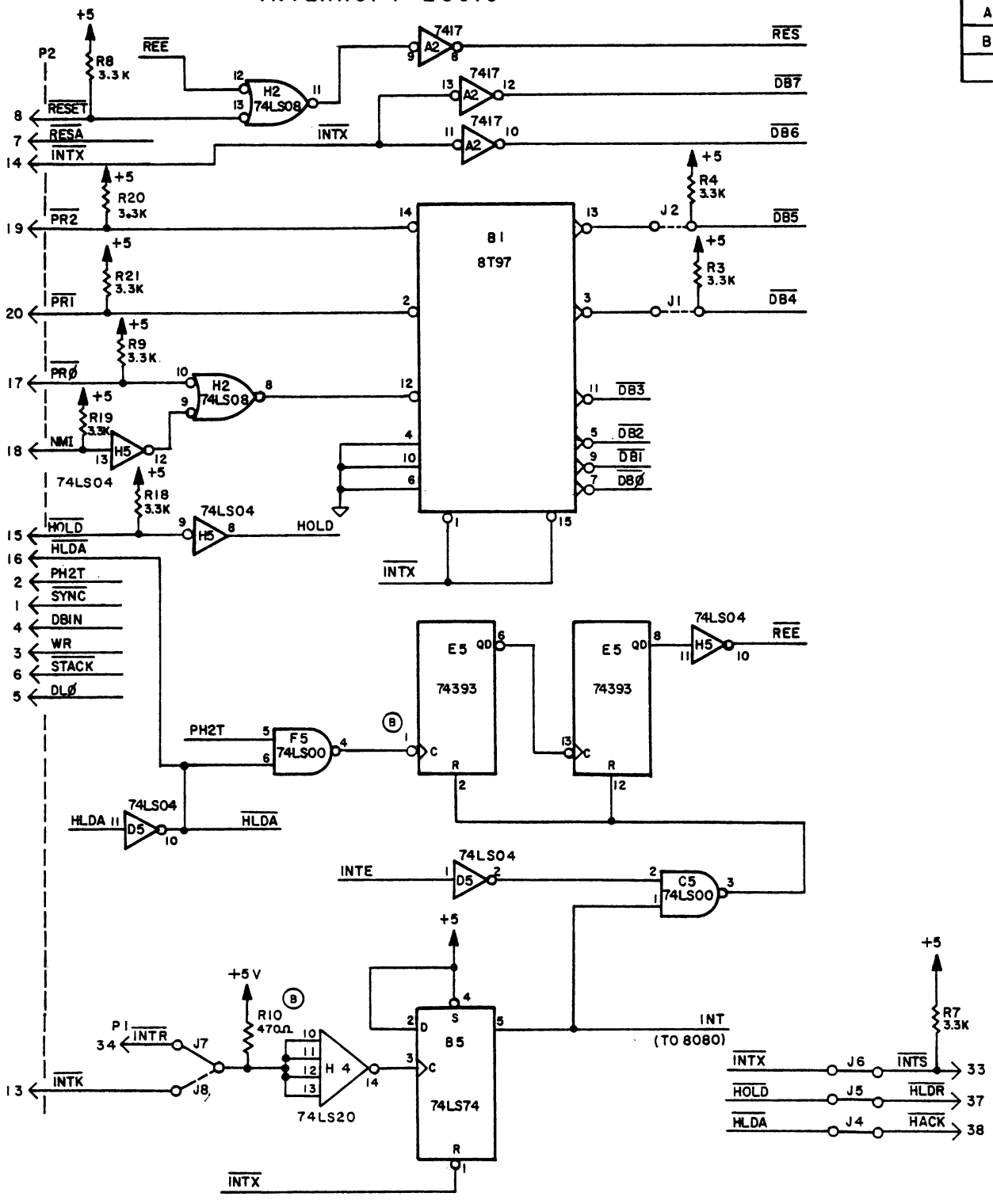

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY D.T.A. | DRAWING NO. DS 471 PAGE 6 |
|---|---|---|---|
| | DATE 12-22-75 | DATE 1-8-76 | |

EF4A

modular micros martin research

| REV |
|---|
| A |
| B |
| |

## INTERRUPT LOGIC




| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY DM | DRAWING NO. DS 471 PAGE 7 |
|---|---|---|---|
| | DATE 12-22-75 | DATE 1-8-76 | |

EF4A

# CENTRAL PROCESSING UNIT
## MODEL 471

| REV |
|---|
| A |
| |
| |

43 ← +12
44 ←
45 ← −12
46 ← −9
47 ←

## POWER BUSSING

+5

48 ←
49 ←
50 ←

3 LM3302 12
14 7417 7
14 74LS00 7
14 74LS04 7
14 74LS08 7
14 74LS20 7
14 74LS74 7

+ C1 22µf MIN.

C3, 4, 6, 7, 9, 11, 13, 14, 16 .05µf MIN.

16 74LS174 8
14 74393 7
16 8224 8
16 8T26 8
16 8T97 8
14 9601/74122 7

→ 39
→ 40
→ 41
→ 42

| | | | |
|---|---|---|---|
| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY DTA | DRAWING NO. DS 471 PAGE 8 |
| | DATE 12-22-75 | DATE 1-8-76 | |

EF4A

modular micros martin research

SEC. 1.3 *BUS DRIVERS* (cont'd.)

In the *modular micro* series, the 16 address lines are designated DH7 through DHØ (corresponding to A15 through A8 on the 8080 package), and DL7 through DLO (A7 through AØ).

The 8T26s can sink up to 40 mA of current (about 25 standard TTL loads), and the 8T97s, up to 48 mA (about 30 standard TTL loads)--more than enough for most microcomputer systems. In addition, both bus driver types are assured of delivering a logic high input voltage of at least +3.5 V to the 8080 (at rated 8080 leakage currents), meeting the 8080 $V_{IH}$ specification of +3.3 V.

1.4 *Main Timing* The 8080 requires a two-phase clock referenced to +12 V. The standard 8080 requires a clock cycle time, $t_{CY}$, ranging between 480 ns and 2.0 us; the 471 runs with a 500 ns clock cycle time, for an instruction cycle time of 2.0 us. An *instruction cycle* is defined for speed spec purposes as the time required for the 8080 to execute an instruction consisting of a single *machine cycle* consisting of four *states*, each state lasting one clock period. Actual instruction cycles vary in duration between four and 18 clock periods, depending on the instruction involved. The number of clock periods is further increased if the CPU is made to wait for slow memory.

The 471 CPU board uses a type 8224 clock generator/driver for supplying the 8080 with the required PH1 and PH2 signals. An 18.0 MHz crystal is connected to the 8224, whose internal oscillator produces an 18 MHz square wave (available at pin 12 for test purposes). An internal divide-by-nine counter produces two alternating positive pulses with a basic frequency of 2.0 MHz and an asymmetrical duty cycle (see Fig. 1.4.1). After passing through high-level drivers, these signals are connected directly to the 8080, as PH1 and PH2. A TTL version of PH2--PH2T--is also generated, and is available for system control purposes.

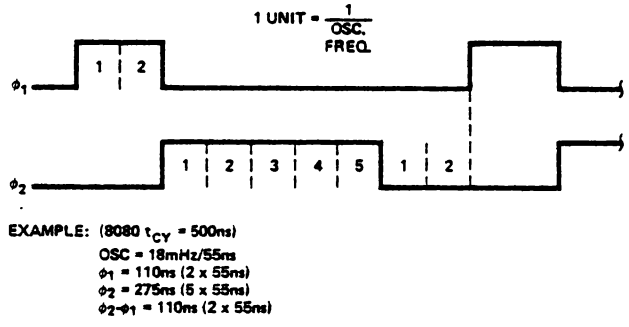



*Fig. 1.4.1--Asymmetrical 8080 Clock*

At press time, available high-speed versions of the 8080A (with maximum clock periods shown in nanoseconds) included Intel C8080A-2, Advanced Micro Devices Am9080A-2 (380); C8080A-1, Am9080A-1 (320); Am9080A-4 (250). The Intel 8224 can be used to generate 8080 clock signals up to a maximum frequency of 27 MHz (corresponding to a clock period down to 370 ns), which supports the 8080A-2. The Am8224 is specified to function with CPUs up to the 250 ns clock cycle speed. (These figures were those available at press time; new product releases were scheduled, and the reader should check for current specifications.)

SEC. 1.4 *MAIN TIMING* (cont'd.)

In order to permit the use of an overtone crystal which may be more economical at frequencies above 20 MHz--a tank circuit may be added, as shown by the optional parts connected to the 8224 in the 471 schematic. L1 and C12 are chosen for resonance at the overtone frequency, and C1 is an RF bypass capacitor.

The 8224 also supports three system control functions. First, it generates $\overline{STB}$, a *status strobe* occurring at the beginning of each 8080 machine cycle. $\overline{STB}$ is used to strobe system control logic (Sec. 1.5), and is made up of SYN (an 8080 signal indicating the beginning of a machine cycle) and $\phi$1A, which occurs "1 unit" before $\phi$1 proper (referring to Fig. 1.4.1). Using $\phi$1A and incorporating this logic in the Schottky bipolar circuitry of the 8224 optimizes system speed. The 8224 circuitry also activates $\overline{STB}$ during reset cycles.

The 8224 synchronizes the system reset signal at its $\overline{RES}$ input with $\phi$2D of the clock (the rising edge of $\phi$2), to provide the synchronization required by the 8080. (For more on the reset function, see Sec. 1.7 below.)

Finally, the 8224 synchronizes the microcomputer's $\overline{BSY}$ line with $\phi$2D of the clock, again to meet 8080 internal timing requirements. (For more on use of the $\overline{BSY}$ line for memory wait cycles, see Sec. 1.10.)

1.5 *System Control Logic* The 8080 is a synchronous device which follows certain intricate sequences in executing program instructions stored in memory. An *instruction cycle* is required for the execution of each instruction, with this time period subdivided into specialized cycles, as required in order to time-share the microprocessor's eight-bit data bus and sixteen-bit address bus. An instruction cycle consists of between one and four *machine cycles*, with the first always devoted to *fetching* the computer's progrom instruction from memory. The address bus selects the instruction location, and the data bus receives the memory word. In general, one additional machine cycle is necessary each time the instruction writes into or reads from memory or an I/O port. (For detailed information on processor cycles, the reader is referred to the 8080 data sheet, DS2204.)

Near the beginning of each machine cycle, the 8080 produces the signal SYNC, which is gated by the 8224 clock generator to produce the system strobe STB. At STB time, the 8080 data bus contains an eight-bit *status word* which defines the purposes of the machine cycle. The assignment of data bus bits is as follows:

| | | |
|---|---|---|
| D7 | MEMR | Memory to be read from |
| D6 | INP | Input data to be received |
| D5 | $M_1$ | First byte of instruction to be fetched |
| D4 | OUT | Output data to be transmitted |

modular micros martin research

SEC. 1.5 *SYSTEM CONTROL LOGIC* (cont'd.)

| | | |
|---|---|---|
| D3 | HLTA | CPU in the HALT state |
| D2 | STACK | Stack operation |
| D1 | $\overline{WO}$ | Write into memory or output port (active low) |
| DØ | INTA | CPU acknowledges an interrupt request |

The system control logic on the 471 CPU board includes a 74LS174 hex latch and a 74LS74 D flip-flop. Both circuits are strobed by STB ($\overline{STB}$ inverted), latching up seven of the eight bits of the 8080 status word. (The $M_1$ bit is not used.) Note that the inputs to the latches derive from the inverted data bus, $\overline{DB7}$ through $\overline{DB\emptyset}$; the data terminals of the latches have been bubbled to indicate the active-low logic involved. Three of the resultant outputs are inverted and gated with DBIN to produce $\overline{MEMRA}$, $\overline{IORA}$, and $\overline{INTX}$. DBIN comes from the 8080 (through buffering inverters), and indicates that the 8080 data bus terminals are ready to receive from external circuitry. $\overline{INTX}$ is used to read in an interrupt jam instruction, during interrupt cycles (Sec. 1.8). $\overline{MEMRA}$ and $\overline{IORA}$ are buffered by the Address/Control Bus Driver stage (Sec. 1.3) and are relabeled $\overline{MEMR}$ and $\overline{IOR}$. Distributed via the *modular micro* bus to circuitry outside the 471 CPU board, these signals are used for enabling memory to be read from, and input ports to be read from (respectively).

The 8080 creates an enable signal called $\overline{WR}$, indicating that it has valid data on the data bus to be written either into memory or into an output port. To distinguish between these write operations, two NAND gates combine WR ($\overline{WR}$ inverted) with the status word bit OUT, which is latched up in the 'LS74 and is available both true and complemented at the flip-flop outputs. $\overline{STEWA}$ (sometimes called $\overline{MEMWA}$) is active during memory write cycles, and $\overline{IOWA}$ during output write cycles. Both are buffered by the Address/Control Bus Drivers and appear on the *modular micro* bus as $\overline{STEW}$ and $\overline{IOW}$. Note that these are enable signals, valid only at the moment when the 8080 is transmitting data on its data bus.

The signal $\overline{MEMA}$ is produced by ORing together $\overline{MEMRA}$ and $\overline{STEWA}$, and after buffering, is available as $\overline{MEM}$ on the *modular micro* bus. It is a combined strobe/enable signal, active during cycles referring to memory (read or write), at the moment when data is transferred between CPU and memory via the data bus.

The signal $\overline{PCWA}$ indicates that a memory write cycle is in progress. Unlike the signals described above, it is not a strobe/enable signal--that is, it is *not* valid just at the moment when data is transferred on the data bus. Rather, $\overline{PCWA}$ starts near the beginning of a memory write cycle and does not end until the next machine cycle begins. A NAND gate is used to gate WO (a status bit indicating a memory or output write cycle) with $\overline{OUT}$, in such a way that $\overline{PCWA}$ is active only when $\overline{OUT}$ is inactive. After buffering, the signal is called $\overline{PCW}$, and is available on the *modular micro* bus. Typically, $\overline{PCW}$ is used to control the R/$\overline{W}$ terminals of RAM memory. $\overline{PCW}$ anticipates both $\overline{STEW}$ and $\overline{MEMR}$, so that the read/$\overline{write}$ mode on the RAM can be set up before the strobe/enable signal activates data transfers with the CPU.

SEC. 1.5 *SYSTEM CONTROL LOGIC* (cont'd.)

Another signal useful in interfacing with memory is the status strobe, $\overline{STB}$, developed by the 8224 clock generator and discussed above. After buffering by the Address/Control Bus Driver stage, it is piped along the *modular micro* bus under the label $\overline{ADV}$, for *address valid.* As the name implies, when $\overline{ADV}$ occurs the 8080A address lines have been loaded and are stable, indicating that the address bus may be used reliably in memory and I/O addressing. $\overline{ADV}$ occurs early enough in the machine cycle that, should the ready line be pulled down at this time, the CPU will enter a wait state. (See Sec. 1.10.)

1.6 *I/O Instruction Modes*

1.6.1 *Standard 8080 I/O Mode* The $\overline{IOR}$ and $\overline{IOW}$ signals discussed above are used for strobing and enabling input and output ports, using the standard 8080 INP and OUT instructions. In either of these instructions the second byte is an eight-bit I/O port address, which appears on both the DH and the DL address lines of the microcomputer. The I/O circuitry must first be addressed by the DH or DL address lines; then the $\overline{IOR}$ signal enables an input port (or $\overline{IOW}$ strobes an output port) and the I/O data flows between the port and the 8080 accumulator (A register).

1.6.2 *Memory-Mapped I/O* A second mode of I/O addressing is commonly used--simply defining selected addresses in memory as I/O ports. Output ports are addressed by memory write instructions--strobed with the $\overline{STEW}$ ($\overline{MEMW}$) strobe. Input ports are addressed by memory read instructions--enabled with $\overline{MEMR}$ signals. This *memory-mapped* I/O mode is that used in the 6800/6502 microprocessor series, and is attractive because of its universality. That is, the approach is basically compatible with any microcomputer, and a peripheral device with this kind of I/O port addressing circuitry can most likely be interfaced with few modifications to different microprocessors. This is especially true in the *modular micro* series, where the bus is compatible with various standard eight-bit processors. Another advantage is that the 8080A includes instructions for transferring data between memory and a number of CPU index registers, rather than just the A register (accumulator) as with I/O instructions proper. Since the computer's memory read/write signals are used in controlling memory-mapped I/O, no additional circuitry is required. Naturally the user must maintain a memory map, to make sure that memory and I/O port addresses do not overlap.

By predecoding some of the I/O address lines, producing a specialized I/O strobe/enable signal, the amount of I/O strobe decoding circuitry necessary on peripheral boards in the microcomputer can be reduced. (See Sec. 1.6.4.)

1.6.3 *8008-Compatible I/O* A third mode of I/O addressing is supported by the 471 CPU board, which derives the $\overline{DINA}$ strobe/enable signal (as shown in the System Control Logic section of the schematic). This I/O addressing mode is used by the 8008 microprocessor, the 8080's predecessor, widely used in industrial controllers. An 8008 microcomputer--the *MIKE 2*--is available from Martin Research, and some boards in the *modular micro* series use the 8008 mode of I/O addressing. The fact that the 471 includes these features

SEC. 1.6 *I/O INSTRUCTION MODES* (cont'd.)

| I/O ADDRESS BITS | | | | | MEMORY PAGE USING $\overline{DIN}$ | | EQUIVALENT 8008 INSTRUCTION | | MEMORY PAGE USING $\overline{DIN2}$ | |
|---|---|---|---|---|---|---|---|---|---|---|
| DH5 | DH4 | DH3 | DH2 | DH1 | OCTAL | HEX | MNEMONIC | OCTAL CODE | OCTAL | HEX |
| 0 | 0 | 0 | 0 | 0 | 301 | C1 | INP 00 | 101 | 300 | C0 |
| 0 | 0 | 0 | 0 | 1 | 303 | C3 | INP 01 | 103 | 302 | C2 |
| 0 | 0 | 0 | 1 | 0 | 305 | C5 | INP 02 | 105 | 304 | C4 |
| 0 | 0 | 0 | 1 | 1 | 307 | C7 | INP 03 | 107 | 306 | C6 |
| 0 | 0 | 1 | 0 | 0 | 311 | C9 | INP 04 | 111 | 310 | C8 |
| 0 | 0 | 1 | 0 | 1 | 313 | CB | INP 05 | 113 | 312 | CA |
| 0 | 0 | 1 | 1 | 0 | 315 | CD | INP 06 | 115 | 314 | CC |
| 0 | 0 | 1 | 1 | 1 | 317 | CF | INP 07 | 117 | 316 | CE |
| 0 | 1 | 0 | 0 | 0 | 321 | D1 | OUT 10 | 121 | 320 | DO |
| 0 | 1 | 0 | 0 | 1 | 323 | D3 | OUT 11 | 123 | 322 | D2 |
| 0 | 1 | 0 | 1 | 0 | 325 | D5 | OUT 12 | 125 | 324 | D4 |
| 0 | 1 | 0 | 1 | 1 | 327 | D7 | OUT 13 | 127 | 326 | D6 |
| 0 | 1 | 1 | 0 | 0 | 331 | D9 | OUT 14 | 131 | 330 | D8 |
| 0 | 1 | 1 | 0 | 1 | 333 | DB | OUT 15 | 133 | 332 | DA |
| 0 | 1 | 1 | 1 | 0 | 335 | DD | OUT 16 | 135 | 334 | DC |
| 0 | 1 | 1 | 1 | 1 | 337 | DF | OUT 17 | 137 | 336 | DE |
| 1 | 0 | 0 | 0 | 0 | 341 | E1 | OUT 20 | 141 | 340 | EO |
| 1 | 0 | 0 | 0 | 1 | 343 | E3 | OUT 21 | 143 | 342 | E2 |
| 1 | 0 | 0 | 1 | 0 | 345 | E5 | OUT 22 | 145 | 344 | E4 |
| 1 | 0 | 0 | 1 | 1 | 347 | E7 | OUT 23 | 147 | 346 | E6 |
| 1 | 0 | 1 | 0 | 0 | 351 | E9 | OUT 24 | 151 | 350 | E8 |
| 1 | 0 | 1 | 0 | 1 | 353 | EB | OUT 25 | 153 | 352 | EA |
| 1 | 0 | 1 | 1 | 0 | 355 | ED | OUT 26 | 155 | 354 | EC |
| 1 | 0 | 1 | 1 | 1 | 357 | EF | OUT 27 | 157 | 356 | EE |
| 1 | 1 | 0 | 0 | 0 | 361 | F1 | OUT 30 | 161 | 360 | FO |
| 1 | 1 | 0 | 0 | 1 | 363 | F3 | OUT 31 | 163 | 362 | F2 |
| 1 | 1 | 0 | 1 | 0 | 365 | F5 | OUT 32 | 165 | 364 | F4 |
| 1 | 1 | 0 | 1 | 1 | 367 | F7 | OUT 33 | 167 | 366 | F6 |
| 1 | 1 | 1 | 0 | 0 | 371 | F9 | OUT 34 | 171 | 370 | F8 |
| 1 | 1 | 1 | 0 | 1 | 373 | FB | OUT 35 | 173 | 372 | FA |
| 1 | 1 | 1 | 1 | 0 | 375 | FD | OUT 36 | 175 | 374 | FC |
| 1 | 1 | 1 | 1 | 1 | 377 | FF | OUT 37 | 177 | 376 | FE |

*Figure 1.6.1--8080 Memory Pages for 8008 I/O Instructions*

modular micros ⇄ martin research

SEC. 1.6.3 *8008-COMPATIBLE I/O* (cont'd.)

makes it very attractive in updating 8008 systems, especially when it is not desired to change peripheral I/O boards.

In the 8008 I/O mode, the I/O port address appears on five lines of the high-order memory address bus, namely DH5 through DH1. *Output* data appears on the low-order memory address, DL7-DLØ. *Input* data is transmitted to the CPU via the data bus. When executed on an 8008, there are special single-byte input and output instructions; but with the Model 471 8080 CPU board, these 32 I/O ports are addressed with certain memory referencing instructions.

Referring to the circuitry which creates $\overline{\text{DINA}}$: this signal occurs when the 8080 data bus is ready to receive data ($\overline{\text{DBIN}}$) or transmit data ($\overline{\text{WR}}$); when the instruction is not a standard 8080 I/O instruction ($\overline{\text{OUT}}$ and $\overline{\text{IN}}$ must be high); and only when DH7 and DH6 are high, and DHØ is high. These conditions mean that any 8080 memory read/write instruction which references the odd-numbered pages in the upper one-fourth of memory can be used as an 8008-compatible I/O instruction. (See Figure 1.6.1.)

$\overline{\text{DIN2A}}$ is a variant of $\overline{\text{DINA}}$, differing only in that it is created when DHØ is low (rather than high). By using $\overline{\text{DIN2}}$, another set of 32 I/O ports can be addressed. Again, see Fig. 1.6.1.

The Address/Control Bus Driver stage converts $\overline{\text{DINA}}$ to $\overline{\text{DIN}}$, and $\overline{\text{DIN2A}}$ to $\overline{\text{DIN2}}$, for use on the *modular micro* bus. Note that the user must choose between the extra 32 8008-compatible I/O ports (using the $\overline{\text{DIN2}}$ signal), and the standard 8080 input ports (using the $\overline{\text{IOR}}$ signal), by wiring up *either* jumper 9 (J9) *or* J10. As the solid line indicates, factory-assembled boards are wired for $\overline{\text{IOR}}$.

To decode the strobe and enable signals for an 8008-compatible I/O port, circuitry on a peripheral board combines the $\overline{\text{DIN}}$ or $\overline{\text{DIN2}}$ signal on the *modular micro* bus with the five address bits DH5 through DH1. Using a single 74LS138 (3205/8205), eight such strobe/enable signals can be produced by connecting the five DH bits in various combinations to the three select inputs, and to one active-low and one active-high enable input. (This technique is elaborated in the data sheet for a Universal I/O Kit, KT9001, and in Chapter 7 of *Microcomputer Design.*)

Several kinds of 8080 instructions may be performed using the 471 CPU board in order to address 8008-compatible I/O ports. These are indicated in Figure 1.6.2.

SEC. 1.6 *I/O INSTRUCTION MODES* (cont'd.)

| I/O PORT ADDRESS IS IN: | OUTPUT DATA IS IN: | 8080 INSTRUCTION EXECUTED: | | OUTPUT DATA ENDS UP IN: | INPUT DATA ENDS UP IN: |
|---|---|---|---|---|---|
| H | L | MOV | A, M | OUTPUT PORT | A |
| B | C | LDAX | B | OUTPUT PORT | A |
| D | E | LDAX | D | OUTPUT PORT | A |
| 2ND INST. BYTE (PPP) | 3RD INST.BYTE (DDD) | LDA | PPPDDD | OUTPUT PORT | A |
| 2ND INST. BYTE (PPP) | 3RD INST.BYTE (DDD) | STA | PPPDDD | OUTPUT PORT | NO INPUT-- A REG. IS UNCHANGED |

*Figure 1.6.2--8080 Instructions Referencing 8008-Compatible I/O Ports*

Note that many of these 8080 instructions can be used as *combined I/O* instructions. Assume that a strobe decoder on a peripheral I/O board combines $\overline{DIN}$ with some combination of DH5-DH1--say 00100, corresponding to octal memory page 311 (Fig. 1.6.1). The resulting signal is used to strobe an eight-bit latch, whose inputs are connected to the DL address bus. This is an output port--equivalent to 8008 Output Port 4. The same signal is used to enable an eight-bit bus driver, with inputs deriving from some eight-bit parallel data source, and tri-state driver outputs connected to the microcomputer data bus (8008 Input Port 4). The 8080 instruction LDA 311DDD is executed (where DDD is the output data). $\overline{DIN}$ goes low; the H address 311 is decoded; both input and output ports are activated; the output data DDD ends up in the output port; and the input data is moved to the A register.

For example, the program . . .

```
LXI  B,311123
LDAX B
```

. . .will output the octal value 123 to Output Port 4, and simultaneously load the A register with the data at Input Port 4. The instruction LDA 311123 would accomplish the same thing.

Note also that if the purpose of the instruction is to *input* data only, the low-order address need not set up. For example, the instruction LDAX B might be executed, with the input port address previously loaded into the B register, but any arbitrary data in the C.

modular micros martin research

SEC. 1.6 *I/O INSTRUCTION MODES* (cont'd.)

Finally, note that the use of these 8008-compatible I/O instructions is entirely optional. Any of the page addresses shown in Fig. 1.6.1 may instead be used for memory, activated by the normal 8080 memory signals ($\overline{STEW}$, $\overline{MEMR}$, etc.), and $\overline{DIN}$ may be ignored. As with other forms of memory-mapped I/O--of which this is a specialized version--the only constraint is that a given memory page address not be used for more than one purpose (memory or I/O port).

1.6.4 *Memory-Mapped I/O with Predecoded Strobe* One advantage of the 8008-compatible I/O mode discussed above is that there are only five I/O address lines, as opposed to eight with the 8080 mode and eight with the memory-mapped mode (if a whole page in memory is used for each I/O port; 16 lines if a single location in memory is assigned as an I/O port). This means that circuitry needed to produce I/O strobes, for activating input bus drivers and output latches on a peripheral circuit board, must combine eight or more I/O address lines, plus a control signal ($\overline{IOR}$, $\overline{IOW}$, $\overline{MEMR}$, $\overline{STEW}$). This circuitry will be cumbersome, especially if more than one I/O strobe/enable signal is needed on that board.

The user of the 471 CPU board can combine the advantages of several I/O address modes by using $\overline{DIN}$ (or $\overline{DIN2}$) for memory-mapped I/O. To output data, the program executes a memory write instruction, and data is written into the output port; to input data, the program executes a memory read instruction, and data is read from the input port into the CPU. All of the various memory-referencing instructions of the 8080 may be used. So far, this is a description of memory-mapped I/O in general. The only differences are these: the I/O ports use $\overline{DIN}$ (or $\overline{DIN2}$) instead of $\overline{MEMR}$ or $\overline{STEW}$; and the high-order memory address which references these I/O ports must be one which will cause $\overline{DIN}$ (or $\overline{DIN2}$) to be produced. (See Fig. 1.6.1.)

With this I/O method, a single 74LS138 on a peripheral board will produce eight I/O strobe/enable signals, as described in Sec. 1.6.3 above.

1.7 *Reset Circuitry* When power is first applied to the 8080, its internal registers may take on any random values. To initialize the machine, the 8080 RESET terminal must be brought to a logic high level for at least three clock cycles. This resets the program counter to zero, and the 8080 will proceed to execute stored program instructions, starting at location zero. The reset signal also resets the 8080 interrupt flip-flop, disabling interrupts (as does an interrupt itself). An initial interrupt is unnecessary for starting the computer. However, if it is desired to make the CPU wait for an initial interrupt before proceeding with its mainline program, the two instructions EI (enable interrupt), HLT (halt) may be inserted starting at location zero. 8080 flags and registers other than the program counter and interrupt flip-flop are not affected by the reset pulse, but must be initialized as required through program instructions.

The Model 471 CPU board includes a Power On Reset circuit which senses the rising of the +5 volt power supply as power is applied, generating the signal $\overline{RES}$. The LM3302/339 is capable of operation at supply voltages as low as +2.0 V. Until the +5 volt line reaches +3.9 V, the zener diode does not conduct, and the - input to the comparator receives the full power supply voltage.

SEC. 1.7 *RESET CIRCUITRY* (cont'd.)

The resistive voltage divider, R13/R11, provides the + comparator input with 9/10 of the power supply voltage; and since the – input exceeds the + input level, the comparator output goes low, activating $\overline{RES}$. When the +5 volt line reaches about +4.3 volts, the voltage at the + comparator input finally begins to exceed the 3.9 V zener voltage, and the comparator goes high, remaining so as the +5 volt supply finishes charging up to a full +5.0 volts, ± 5%.

The $\overline{RES}$ signal is connected to the 8224 clock generator, which synchronizes the reset request with 8080 timing (Sec. 1.4 above). The output of the 8224, RESA, is connected directly to the 8080A CPU.

Should the +5 volt line ever dip below about 4.3 V, the comparator will again go low, after a delay while C8 discharges--generating $\overline{RES}$ and resetting the computer. The delay will prevent short glitches on the power supply from causing a reset. But if the voltage drops much below 4.0 V, the diode (DA806) starts to conduct, discharging the capacitor more rapidly, and causing a reset.

Note that this reset circuit looks only at the +5 volt power supply. For reliable operation of the computer, the +12 V and -12 (or -9) V supplies, also powering the 8080, should come up if not before, at least very soon after the +5 volt power supply does.

An optional reset switch may be installed, allowing the user to reset the microcomputer manually. This provision is generally unnecessary in production versions of the 471 CPU board, but is useful in prototyping. Consider the following potential 8080 system failure: a DI (disable interrupt) instruction has been executed, in order to protect a subroutine which must not be interrupted (such as a critical timing loop or stack operation). Because of a software or hardware bug, a HLT (halt) instruction is executed. This condition is inescapable through interrupts, since the CPU has masked them out. Only a system reset will restart the computer.

As an additional protection against this failure mode, the 471 board incorporates a digital timer which automatically resets the 8080 if the CPU has masked interrupts (through a DI instruction) and fails to acknowledge interrupt requests for a set period. The timer consists of a 74393 dual divide-by-16 counter (see page 7, location C2). The 8080 has an output terminal labeled INTE (interrupt enable) which is high when interrupts are enabled. When the CPU is in the state where interrupts will be acknowledged, the inverted version of INTE causes a low input to a 74LS00 NAND gate, whose output is therefore high. This holds the 74393 reset (R) terminals at a high level, forcing the counter output bits to zero and inhibiting counting.

When interrupts are disabled, INTE goes low. If at the same time, an interrupt request has been received, but has not yet been acknowledged by the CPU, the 74LS74 interrupt request flip-flop output (B5, pin 5) is high. These conditions are combined by the NAND gate, and the 74393 reset signal is removed, allowing the counters to count. Clocked by PH2T, the 74393 counts once each 8080 clock cycle. If the CPU acknowledges a HOLD request (Sec. 1.9), $\overline{HLDA}$ goes low, inhibiting counting--but not resetting the 74393. If the CPU acknowledges an interrupt

SEC. 1.7 *RESET CIRCUITRY* (cont'd.)

request, then the counters are reset. But if 128 PH2T cycles have been clocked, the high-order 74393 output bit (pin 8) goes high. The signal which results (after inversion), $\overline{REE}$, causes a reset.

$\overline{REE}$ is ORed with the signal $\overline{RESET}$, which originates from an optional interrupt control board (Sec. 1.8). A 7417 open-collector buffer is connected to the $\overline{RES}$ line, which may also be pulled low by the power-on reset comparator and the optional reset switch.

The automatic reset circuit may be defeated by removing the 74393 counter from its socket, and connecting a wire jumper between pins 8 and 7 (ground) on the 74393 socket.

1.8 *Interrupt Logic* The 8080 fully supports external interrupt requests. A typical interrupt source is a peripheral input device (such as a keyboard) which inputs data relatively infrequently and unpredictably, but which must receive immediate priority when ready. Basically, the 8080 receives the interrupt request and issues an interrupt acknowledgment. External circuitry detects the acknowledgment, and jams in a special instruction, causing the CPU to jump to the appropriate interrupt subroutine. When this ends, the CPU returns to the mainline program at exactly the point where it was interrupted.

In many systems there are multiple interrupt sources. It is necessary therefore to provide means for decoding the source of an interrupt request, in order to make the CPU execute the appropriate interrupt instructions. The 471 includes on-board provisions for three interrupt levels. In addition, the essential interrupt control signals are routed to a 20-pin connector at the side of the board, shown as P2 on the 471 schematic. An optional interrupt control board may be connected here for expansion to an eight-level prioritized interrupt system.

The interrupt cycle begins when some device within the *modular micro* system pulls the $\overline{INTR}$ line (pin 34) low. (Refer to page 7, location D1.) $\overline{INTR}$ is connected through jumper J7 to a 74LS20 inverter stage. (J8 would replace J7 only if the optional interrupt board were installed.) The 74LS74 interrupt request flip-flop is clocked, causing the INT signal to go high. The INT signal is connected directly to the 8080, requesting an interrupt.

The 8080 contains an internal interrupt flip-flop, INTE (as mentioned above). Executing a DI (disable interrupt) instruction, resetting the computer, or performing an interrupt, will reset the INTE flip-flop and prevent the CPU from acknowledging further interrupts. To enable interrupts, an EI instruction must be executed. This sets the INTE flip-flop and permits the 8080 to recognize interrupts. The 8080 CPU does not itself save interrupt requests which were placed when interrupts were disabled. However, the 471 CPU board does, in the interrupt request flip-flop (a 74LS74: page 7, location D2 in the schematic).

When an interrupt is recognized, the 8080 enters a special interrupt instruction cycle. In the machine status word for the instruction fetch cycle appearing on the data bus at STB time, $\overline{DBO}$ will be 0. The system control logic detects this interrupt acknowledgment, and at $\overline{DBIN}$

SEC. 1.8 *INTERRUPT LOGIC* (cont'd.)

time, produces the signal $\overline{INTX}$ (Sec. 1.5 above). This enable signal is used to jam the special interrupt instruction onto the data bus. It also resets the 74LS74 interrupt request flip-flop.

$\overline{INTX}$ is connected to the inputs of two 7417 open-collector buffers, and to the enable terminals of an 8T97 hex tri-state buffer. The 7417s pull $\overline{DB7}$ and $\overline{DB6}$ to 0, and the 8T97 pulls $\overline{DB3}$ through $\overline{DB0}$ to 0. ($\overline{DB3}$ is pulled low because pin 12 of the 8T97 is normally low--unless the optional interrupt board pulls down the NMI line.) $\overline{DB5}$ and $\overline{DB4}$ are held high by pullup resistors R4 and R3 (which would be replaced by jumpers J2 and J1 only if the optional interrupt board were installed). (B)

These conditions mean that the inverted data bus bits $\overline{DB7}$ through $\overline{DB0}$ read 00110000. Reinverting, the data word seen by the CPU is 11001111, or 317 octal. This is the code for an 8080 RST 1 instruction, a single-byte call instruction which causes the CPU to jump to memory location 000010, pushing the program counter stack. Starting at this location, the user installs a program which contains the instructions necessary to carry out the purposes of the interrupt request. The interrupt routine ends with a RET (return) instruction, popping the PC stack and returning control to the mainline program.

The reader should note that, to avoid having the interrupt routine disturb the mainline program, any registers and flags used both by the interrupt and mainline programs should be saved as the interrupt routine begins (with PUSH instructions) and restored as it ends (with POP instructions).

The device requesting an interrupt may pull down the $\overline{DB5}$ and $\overline{DB4}$ data bus lines at $\overline{DBIN}$ time, altering the interrupt jam instruction as shown in Figure 1.8.1.

| $\overline{DB5}$ | $\overline{DB4}$ | $\overline{DATA}$ | DATA | OCTAL | INSTRUCTION | RESTART ADDRESS |
|---|---|---|---|---|---|---|
| 1 | 1 | 00110000 | 11001111 | 317 | RST 1 | 000010 |
| 1 | 0 | 00100000 | 11011111 | 337 | RST 3 | 000030 |
| 0 | 1 | 00010000 | 11101111 | 357 | RST 5 | 000050 |
| 0 | 0 | 00000000 | 11111111 | 377 | RST 7 | 000070 |

*Figure 1.8.1--Interrupt Jam Instructions*

This interrupt system effectively provides three interrupt levels, though there are four possible jam instructions (Fig. 1.8.1). RST 1 is a general-purpose interrupt, where the interrupting device does not affect $\overline{DB5}$ or $\overline{DB4}$. Typically this is a low-priority interrupt. A second interrupt port pulls down $\overline{DB5}$, and a third, $\overline{DB4}$. Other data bus lines could be pulled low but will have no

modular micros
martin research

SEC. 1.8 *INTERRUPT LOGIC* (cont'd.)

effect unless the system is provided with the optional eight-level interrupt board. Since a RST 7 interrupt jam instruction would result when *both* the $\overline{DB5}$ and $\overline{DB4}$ interrupt ports requested interrupts at the same time, it cannot be used unambiguously for a third interrupt port. Note also that if a RST 3, 5, or 7 occurs, it masks the possibility that a RST 1 interrupt might also have been requested.

In order to synchronize circuitry on peripheral boards with the CPU interrupt logic, the $\overline{INTX}$ interrupt jam enable signal is connected through jumper J6 to $\overline{INTS}$ on the *modular micro* bus (page 7, location D3). The device requesting an interrupt uses this signal to enable circuitry which pulls down the $\overline{DB5}$ and $\overline{DB4}$ data bus lines. A suitable circuit using only two ICs appears in Figure 1.8.2. (A kit of the parts shown, plus wirewrap sockets, is available from Martin Research as KT9008.)

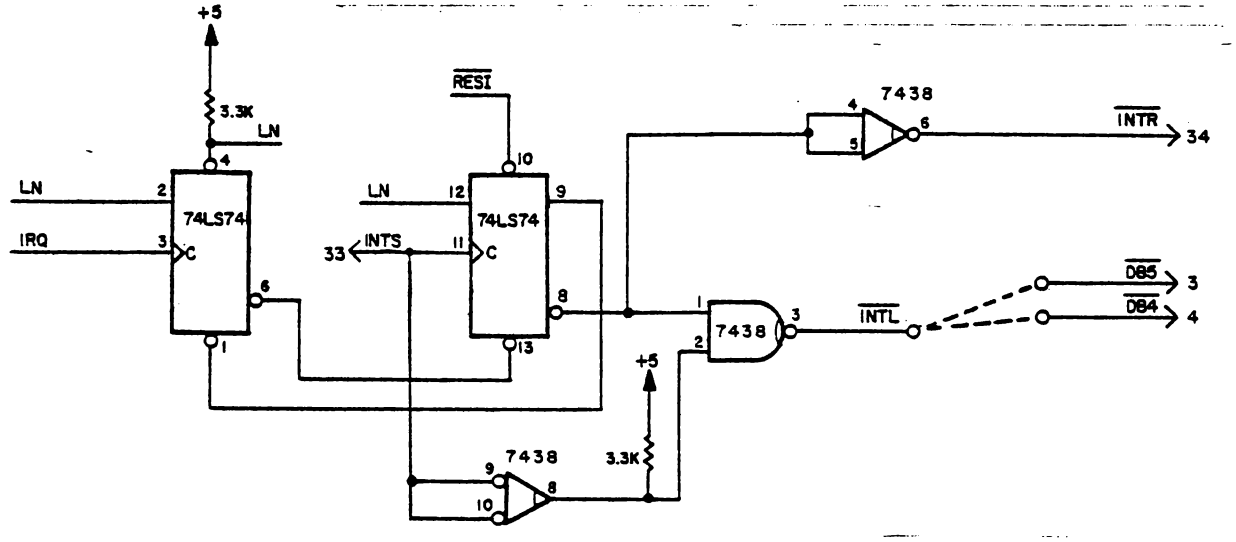



*Figure 1.8.2--Interrupt Request Port*

The device requesting an interrupt causes IRQ (interrupt request) to go high, setting the first 74LS74 flip-flop (pin 3). The $\overline{Q}$ output of this flip-flop (pin 6) resets the second flip-flop (pin 13), causing its Q output (pin 9) to go low. This signal is fed back to the first FF reset terminal (pin 1), resetting the first FF. However, the second FF remains reset. With pin 8 high, the 7438 open-collector buffer connected to the $\overline{INTR}$ line is activated, pulling $\overline{INTR}$ low and requesting an interrupt from the 471 CPU board.

SEC. 1.8 *INTERRUPT LOGIC* (cont'd.)

The 471 acknowledges with an $\overline{INTS}$ enable signal. When $\overline{INTS}$ goes low, it activates a 7438 inverter stage (pins 9, 10, 8). This signal, combined with the high state remaining on the second FF $\overline{Q}$ output (pin 8), activates a 7438 open-collector NAND gate (pins 1, 2, and 3), pulling down $\overline{DB4}$, $\overline{DB5}$, or neither, as required. When $\overline{INTS}$ ends, going high, the second flip-flop is clocked (pin 11) and set. Pin 8 goes low, ending the $\overline{INTR}$ interrupt request, and ending the interrupt jam on the $\overline{DB5}$ or $\overline{DB4}$ data bus line.

The signal $\overline{RESI}$, which sets the second flip-flop (pin 10), will (when low) disable the interrupt request port. If this signal is not used, it should be connected to logic one (LN).

Naturally the device requesting an interrupt would also have other circuitry for communicating with the CPU. If an input device, the restart instruction would cause the CPU to start executing input instructions addressing the appropriate input ports.

1.9 *Hold Circuitry* The 8080 includes provisions for suspending processor operations and allowing some other device to control the microcomputer's data and address busses. Shortly after the 8080 HOLD terminal is brought to a logic high level, the 8080 responds by deactivating its tri-state data and address bus drivers. The 8080 output terminal, HLDA, goes high, acknowledging the hold condition; the inverse of this signal, $\overline{HLDA}$, disables the Address/Control Bus Driver stages. Since the CPU ceases to create $\overline{SYNC}$, $\overline{DBIN}$, or $\overline{WR}$ pulses, the data bus drivers are inactive as well. The 8080 leaves the hold mode when its HOLD input is returned to the logic low condition.

On the 471 CPU board, jumper J5 connects $\overline{HOLD}$ to pin 37 on the *modular micro* bus. Jumper J4 connects the hold acknowledge signal, $\overline{HLDA}$, to pin 38 on the microcomputer bus. (See page 7, location D3.) (These jumpers would be disconnected if the optional priority interrupt board were connected at P2; this board also provides extra DMA channels.)

To initiate a hold condition, a peripheral board pulls $\overline{HLDR}$ low (pin 37). The CPU responds by pulling $\overline{HACK}$ low (pin 38). Special circuitry on the peripheral board is then in control of the address bus, the data bus, and the eight control signals shown in the Address/Control Bus Driver section.

The A version of the 8080 CPU includes internal circuitry for synchronizing a hold request with CPU machine cycles. If an 8080 (original version) is used, the device requesting a hold must do the synchronizing. This is easily done on the peripheral board by gating the hold request circuitry with $\overline{ADV}$ (address valid), pin 26 on the *modular micro* bus.

1.10 *Wait Circuitry* Inspection of the timing diagrams for the 8080 reveals that, operating with a 500 ns clock period, the CPU requires memory with an access time of about 650 ns. However, this is the ideal case, without allowing for delays in bus drivers, address decoders, and other adjunct circuitry in the microcomputer. Intel recommends that the access time of memory interfaced to the 8080 be in the range of 450 to 550 ns, or less. 550 ns is

SEC. 1.10 *WAIT CIRCUITRY* (cont'd.)

probably a safe figure, but the conservative designer may choose to specify memory with a 450 ns access time. 450 ns is Martin Research policy with regard to the Model 405 static 4K RAM board.

If slower memory is used, the 8080 must be made to wait for memory access. This is done by pulling the 8080 READY terminal low. When slow memory is addressed, special memory wait circuitry is triggered, pulling down the $\overline{BSY}$ (busy) line on the *modular micro* bus. To synchronize wait requests with the 8080, $\overline{BSY}$ is connected to the 8224 clock generator, which drives the 8080 READY terminal (Sec. 1.4).

Any peripheral device with slow memory can pull $\overline{BSY}$ down and cause an extended wait period. The same provision may be used with slow I/O devices, during input and output instructions. Optional wait circuitry is a feature of several Martin Research circuit boards.

The 471 CPU board includes a wait circuit consisting of a 9601 or 74122 one-shot and a 7417 open-collector buffer. The one-shot is triggered by STB--a status strobe which comes sufficiently in advance of any read/write data transfer to guarantee that the CPU will enter the wait state. The $\overline{Q}$ output (pin 6) goes low, activating the 7417 and pulling $\overline{BSY}$ low.

The 9601 and 74122 are not identical, but both work in this circuit. They have very similar timing characteristics. The values of R1 and C2 determine the length of the wait cycle. In factory-assembled boards, R1 is 10K, and C2 is 270 pf, producing the required one-shot period of about 915 ns. The goal is to cause two memory wait cycles, which adds two clock cycles (2 x 500 ns = 1.0 us) to the normal T3 state, adding to a total of 1.5 us. This compensates for memory with an access time as slow as 1.55 us.

For this CPU circuitry (8080 with 8224), to cause *n* wait states, the $\overline{BSY}$ line should be pulled low (starting at the leading edge of STB) for the period (n - 1/6) $t_{cy}$, where $t_{cy}$ is the clock cycle time. This formula allows enough latitude to permit using 10% tolerance R and C components.

Greater system speed will result if memory wait cycles occur only when necessary, when slow memory devices are being addressed. It is more efficient to install the wait circuitry on any memory (or I/O) board that requires it, and to use the minimum effective delay in every case. Then the memory wait circuit on the 471 CPU board is deactivated, simply by removing the one-shot from its socket.

1.11 *Indicator Lights* The 471 CPU board has two light-emitting diodes which indicate machine status. A red LED glows when memory wait cycles are being performed; the LED is driven by the 8080 WAIT output terminal, through an inverter and open-collector buffer.

SEC. 1.11 *INDICATOR LIGHTS* (cont'd.)

Another red LED glows when the 8080 is in the halt condition. The signal $\overline{HLT}$ is derived by the system control logic (Sec. 1.5), driving the LED through a 7417. In order to end the halt, an interrupt must be requested (Sec. 1.8) or the machine must be reset. Recall that if the 8080 interrupt flip-flop is reset, it will not recognize interrupts, and only a reset will end the halt condition (Sec. 1.7).

1.12 *Power Requirements* The model 471 board requires +12 V at 75 mA maximum, +5 V at 600 mA max., and -12 V at 7 mA. By removing the 1K resistor at R5 (see the CPU section of the schematic), and installing it as R6, the board may be operated from a -9 V supply rather than -12; the current drawn will be 4 mA.

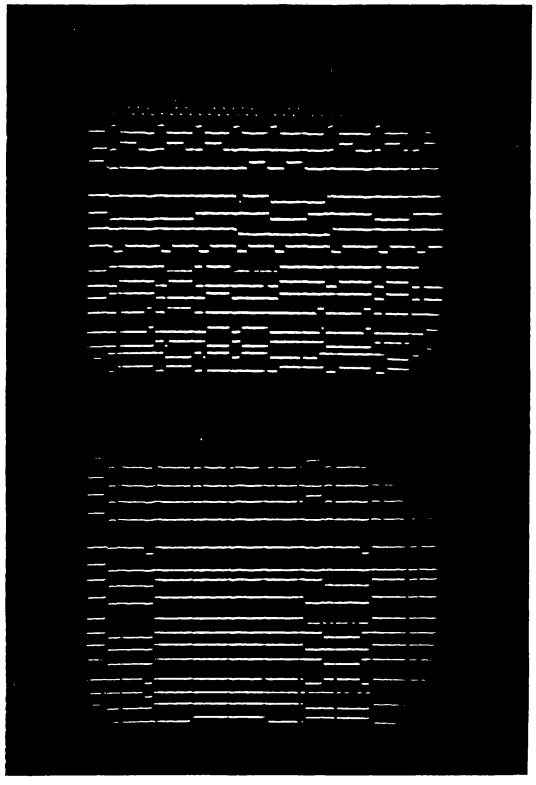
Pins 39 through 50 on the *modular micro* bus are assigned to power supply connections.

REV. A

The *472 DEBUG BOARD* is a special-purpose module in the ***modular micro*** series from Martin Research. Designed for use in development, trouble-shooting, and educational applications, the 472 supports the following:

✓ A ***32-Channel Scope Display*** allows important microcomputer data and timing signals to be displayed on an external single-trace, triggered-sweep oscilloscope. Timing diagrams of data, address, and control lines appear directly on the face of the scope.

*A photograph of an oscilloscope tracing generated by the Model 472 Debug Board appears at the right. Connected to a Model 471 CPU board, based on the 8080 microprocessor, the 32 channels include eight control and eight data lines (top of picture) and the sixteen address lines (bottom of photo).*



✓ A *Delayed Software Interrupt Port* provides a software-generated interrupt request signal--for use in software-controlled single-step debugging of programs.

✓ Also included: sockets for optional pullup resistors on data, address, and control lines--for use in multi-processor, DMA, and interrupt polling schemes. Breadboard space for six 16-pin DIP ICs and one 24-pin DIP.

The Model 472 Debug Board is intended for use with the *MIKE 3* computer, based on the 8080 microprocessor; and with the *MIKE 6* series: either the *MIKE 6.5*, based on the 6502 processor, or the *MIKE 6.8*, based on the 6800. *(The Model 472 is not recommended for use with the NEC upD 8080 processor.)*

The 472 plugs into the system's 50-conductor ribbon cable bus, like other boards in the *modular micro* series. In addition, it connects to the CPU board via a supplementary 20-conductor ribbon cable.

Order information: catalog 472-2 includes an assembled and tested 472 board (less optional pullup resistor packs) plus a 20-pin cable assembly for connection to the system's CPU board. The user must add an additional 50-pin transition connector, part number 1001, to the system's ribbon cable if an open position is not already available. Prices: AT472-2, $89.00; 1001, $6.00. *(U.S. prices, subject to change without notice; standard order terms apply.)* The 472 is also available as part of the *MIKE 3 DEBUG SYSTEM*, model AT814.

REV. A

TECHNICAL DATA

Contents

Sec. 1--Circuit Description

1.1 32-Channel Display
1.2 Delayed Software Interrupt
1.3 Optional Pullup Resistor Packs
1.4 CPU Control Lines

Sec. 2--Parts List

Sec. 3--Application Notes

3.1 Connections to Oscilloscope
3.2 Hints for Best Scope Picture
3.3 Use of Probes (Data Channels 8 and 9)
3.4 Repetitive Reset Mode
3.5 Repetitive Strobe Mode
3.6 The 32-Channel Display in an 8080 System

## SEC. 1 CIRCUIT DESCRIPTION

1.1 *32-Channel Display* The 32-Channel Display allows the user to display microcomputer data bus, address bus, and control line information on an external triggered-sweep oscilloscope. A pushbutton switch located at the left-hand side of the 472 Debug board controls which of two sets of sixteen channels of data will appear on the face of the scope. Normally the scope displays eight control lines (including two user-selected probes) and the eight data bus lines, DB7 through DB0. When the button is pushed, the scope displays the 16 computer address lines, DH7 through DH0 and DL7 through DL0. (For details on scope connections and on interpreting tracings, see Sec. 3 below.)

The display includes two 74150 data selectors, each of which selects among 16 different digital data sources. Referring momentarily to the first 74150 (near the top of the schematic, page 4), this device selects from one of six channels, Data Channels 15 through 10, which derive from the CPU board via the 20-conductor cable; from two probe terminals at the edge of the 472 board; and from the microcomputer data bus, DB7 through DB0. A series of inverters compensates for the inverted state of the

*1.1 32-Channel Display (cont'd)*

of the data bus. The second 74150 selects among the 16 address lines. The active-low outputs of these two data selectors are ORed together; but only one selector output can be active (i.e., low) at once, since the two chips' select (S) inputs are tied to the data/address switch.

Note that external circuitry for switching between the two sets of 16 channels--a contact closure or open-collector gate--may be provided, connecting to the 472 board via the 20-conductor CPU bus, on pin 11.

A 7493 four-bit counter drives both the data selectors and a simple five-bit digital-to-analog converter (DAC). As the counter passes through its 16 output states, 0000 through 1111, it drives both the data selectors--which switch among their 16 input channels--and the DAC. Constituting the four high-order DAC input bits, the counter outputs are buffered through CMOS devices to produce consistent logic high levels. The four resistors, R2-R5, are weighted to position the 16 channel tracings vertically on the oscilloscope screen. The fifth, low-order bit is driven by the data selectors. The DAC output is fed directly to the vertical input of the scope.

Each time that the 7493 is incremented, it selects a different input channel. The trace moves a little higher on the screen, and as the sweep moves across the screen, a timing diagram for the selected channel appears.

The 32-Channel Display requires at least 16 repetitions of a given microcomputer program sequence for the scope to complete its picture. In practice the repetition rate is high, and the eye perceives a single display.

A repetitive signal both drives the 7493 channel counter, and outputs to the oscilloscope External Sync input. This signal may come from one of three sources, at the option of the user.

1) An external oscillator which connects to the 472 board via pin 12 on the 20-conductor CPU bus.

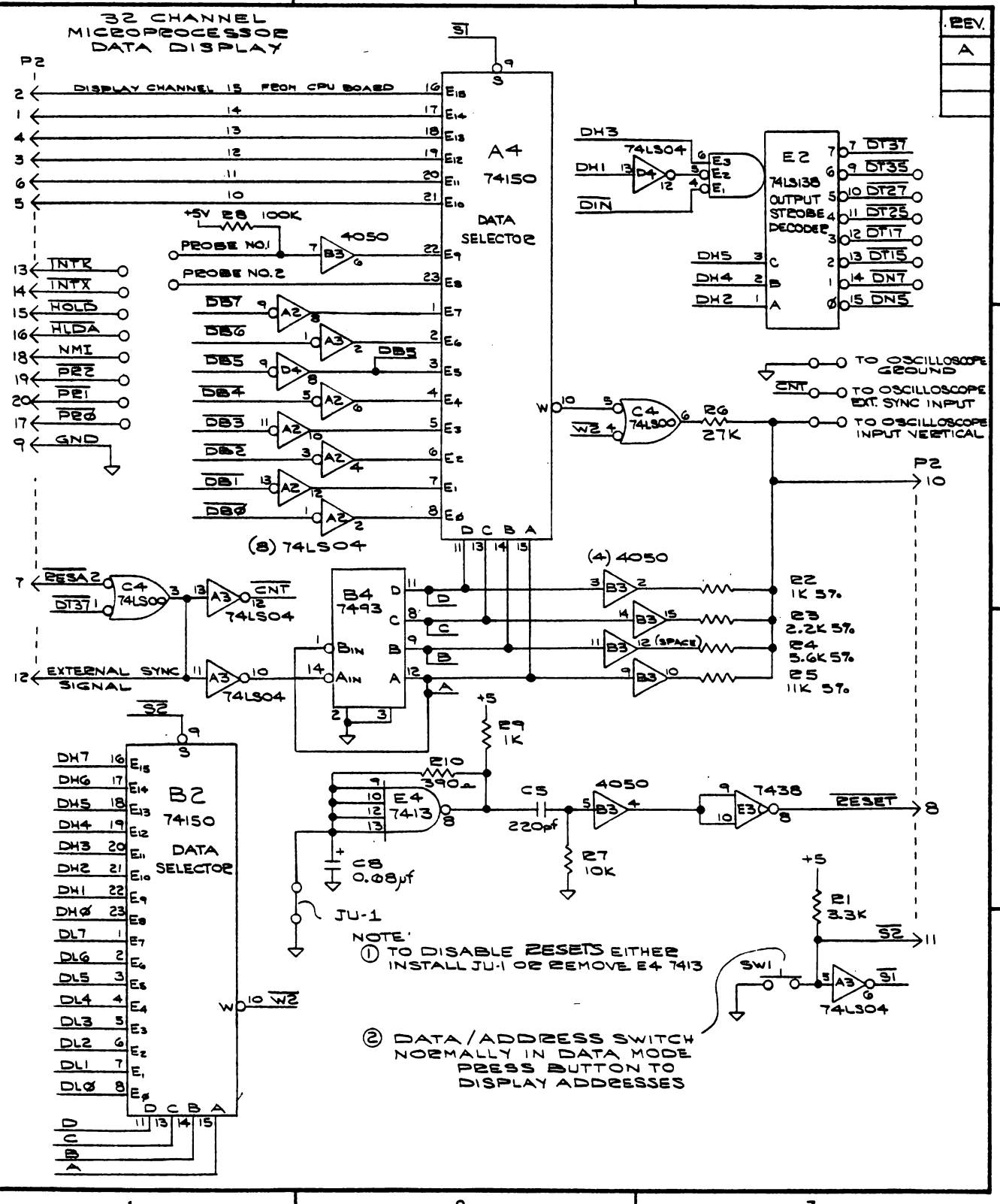
2) A repetitive output strobe, $\overline{\text{DT37}}$, generated by the microcomputer as it executes an OUTPUT 37 instruction contained in the software being debugged.

3) A repetitive reset acknowledge signal, $\overline{\text{RESA}}$, generated (on the CPU board) shortly after it has been reset by a reset oscillator (on the 472 board).

To support the repetitive *output strobe* mode, the 472 board includes a strobe decoder, a 74LS138, which generates $\overline{\text{DT37}}$. To support the repetitive *reset* mode, the board includes a 7413 reset oscillator which repetitively pulls down the $\overline{\text{RESET}}$ line, resetting the CPU. (For further details

# SINGLE STEP & 32 CHANNEL DISPLAY "DEBUG BOARD"

DS472 PAGE 4




| REV. |
|---|
| A |
| |
| |

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R | CHECKED BY | DRAWING NO. DS472 Page 4 |
|---|---|---|---|
| | DATE 3-8-76 | DATE 4-12-76 | |

EF4A

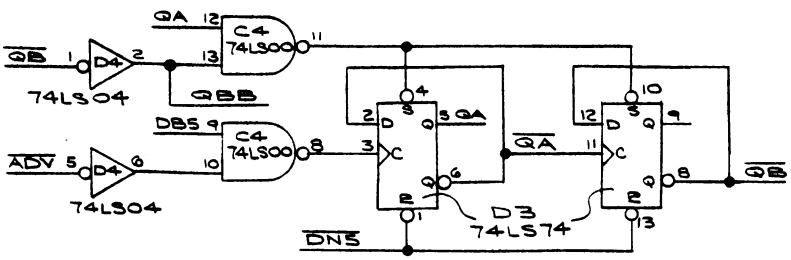
modular micros martin research

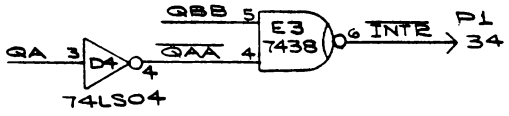
# SINGLE STEP & 32 CHANNEL DISPLAY "DEBUG BOARD"

DS472 PAGE 5

REV. A

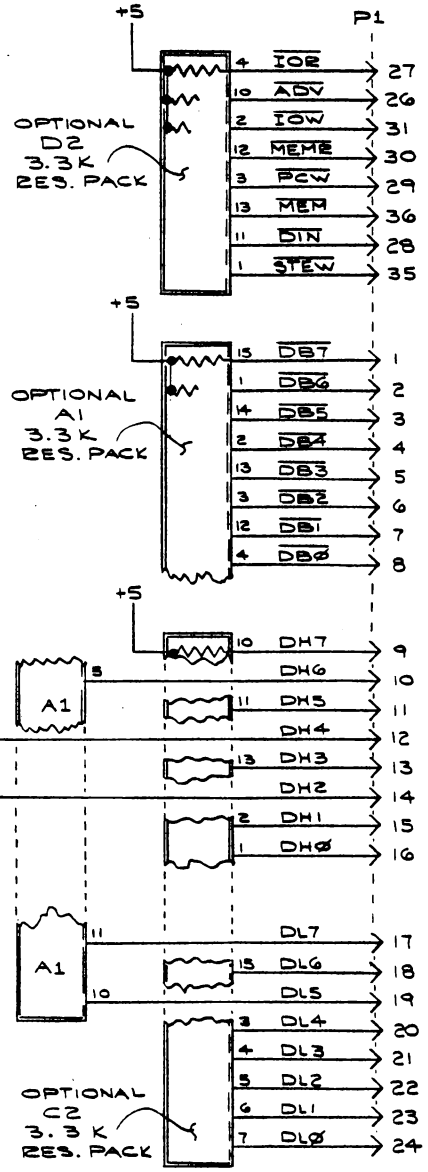
## DELAYED SOFTWARE INTERRUPT CIRCUIT



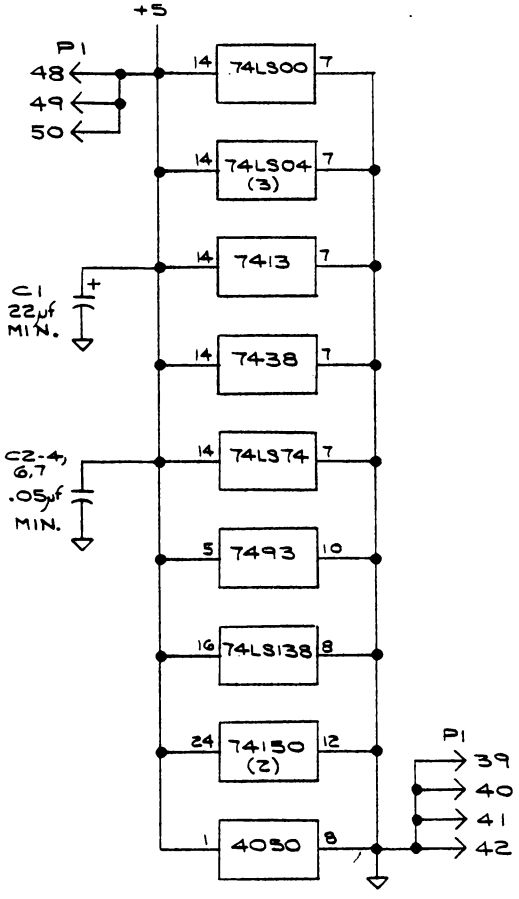




## BUS PULL UP RESISTOR ARRAY

(OPTIONAL, FOR USE WITH DMA; MULTIPROCESSING; PRIORITIZED INTERRUPT SYSTEMS)




## POWER BUSSING




| | | |
|---|---|---|
| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R / DATE 3-8-76 | CHECKED BY D.T.K / DATE 4-12-76 |
| | | DRAWING NO. DS472 Page 5 |

EF4A

modular micros martin research

REV. A

*1.1 32-Channel Display (cont'd)*

on these operating modes, see Secs. 3.3 and 3.4 below.)

The 7413 reset oscillator output is differentiated by an RC combination in order to produce a short pulse, avoiding multiple resets. Buffered by a high-input-impedance CMOS gate, the signal is interfaced to the open-collector reset line via a 7438 gate.

1.2 *Delayed Software Interrupt* It is desirable to be able to cause an interrupt to the microcomputer through the execution of a program instruction in software. It is also useful to have the interrupt request lag behind the execution of the software instruction by a fixed period. The delayed software interrupt circuit on the 472 board is used, in conjunction with suitable programming, to provide such special-purpose functions as single-stepping through programs.

The circuit includes a two-bit counter (74LS74 dual flip-flop) and associated gates, which count instructions following the execution of a designated output instruction. The output strobe, $\overline{DN5}$, is created by a 74LS138 strobe decoder, and occurs when the computer executes an 8008-compatible OUTPUT 5 instruction. In the *MIKE 3*, based on the 8080, this output strobe is generated automatically through the execution of any memory read or write instruction which references page 313 (octal) or page CB (hex) in memory. In the *MIKE 6* series, based on the 6800 or 6502 microprocessors, a memory read or write instruction referencing page 213 (octal) or page 8B (hex) in memory will produce a $\overline{DN5}$ strobe.

$\overline{DN5}$ resets the flip-flops to zero. The first FF is clocked as the computer *address valid* ($\overline{ADV}$) signal ends, going high--but only if a status bit, DB5, is high during $\overline{ADV}$ time, indicating that this is the beginning of the first byte of an instruction. The second time that the first FF is clocked, its output changes and the second FF is also clocked, resulting in the conditions $\overline{QAA}$ = 1; QBB = 1. This causes the 7438 open-collector NAND gate to pull down the interrupt request line, $\overline{INTR}$, requesting an interrupt from the CPU two instructions after the OUTPUT 5 instruction was executed.

During the next instruction, the FFs toggle again. A 74LS00 NAND gate detects the resulting condition and sets both FFs, freezing them until another $\overline{DN5}$ strobe overrides, and the sequence begins again.

1.3 *Optional Pullup Resistor Packs* The 472 board contains sockets for three optional resistor packs. These ICs place 3.3 K pullup resistors to +5 V on the microcomputer address bus, data bus, and control lines. They are useful in interrupt polling schemes, multiprocessor architecture, and DMA circuitry. The purpose is to ensure that all these bus lines go to a logic high level when the CPU and other control devices have been disabled (i.e., tri-state bus-driving circuitry is in the high-impedance state).

1.4 *CPU Control Lines* A 20-conductor ribbon cable between the 472 board and the CPU board carries specialized processor signals to supplement those present on the system's main 50-conductor bus. The 20-pin header, on the 472 board, is shown as P2 on the 472 schematic diagram. Six of these CPU signals are connected to the 32-Channel Display. These signals may vary depending on the CPU board in use; see Sec. 3 and the data sheet for the CPU board for specific information.

| Conductor | Signal |
|---|---|
| 1 | Data channel #14 to scope display (second from top) |
| 2 | Data Channel #15 to scope display (top trace) |
| 3 | Data Channel #12 |
| 4 | Data Channel #13 |
| 5 | Data Channel #10 |
| 6 | Data Channel #11 |
| 7 | $\overline{\text{RESA}}$, reset acknowledge signal from CPU board |
| 8 | RESET, reset request signal to CPU board |
| 9 | Ground, for use with vertical input to scope |
| 10 | Signal to oscilloscope vertical input terminal |
| 11 | Data/address display switch line: pulling low displays addresses; releasing displays data |
| 12 | Signal to oscilloscope external sync input terminal |
| 13 | $\overline{\text{INTR}}$, interrupt request to CPU board |
| 14 | $\overline{\text{INTX}}$, interrupt acknowledge signal from CPU |
| 15 | $\overline{\text{HOLD}}$, hold request to CPU board (floats buses) |
| 16 | $\overline{\text{HLDA}}$, hold acknowledge from CPU board |
| 17 | $\overline{\text{PRØ}}$, interrupt level requested by priority interrupt circuit: low-order bit |
| 18 | $\overline{\text{NMI}}$, non-maskable interrupt request signal to CPU board |
| 19 | $\overline{\text{PR2}}$, interrupt level requested, high-order bit |
| 20 | $\overline{\text{PR1}}$, interrupt level requested, middle-order bit |

REV. A

SEC. 2 *PARTS LIST*

The following parts are included with AT472-2, an assembled and tested Model 472 DEBUG board (less optional resistor packs) with a length of 20-conductor cable and connectors for inclusion in a *modular micro* system.

| Part No. | Quantity | Description |
|---|---|---|
| BD472 | 1 | PC BOARD, DEBUG |
| 1000 | 1 | CONNECTOR, PC board header, 50-pin, rt. angle solder leads |
| 1006 | 2 | CONNECTOR, PC board header, 20-pin, rt. angle solder leads *(one is for associated CPU board)* |
| 1007 | 2 | CONNECTOR, 20-pin transition, clamps onto ribbon cable |
| 1012 | 8 | SOCKET, 14-pin DIP, solder-tail, tin |
| 1013 | 5 | SOCKET, 16-pin DIP, solder-tail, tin |
| 1017 | 2 | SOCKET, 24-pin DIP, solder-tail, tin |
| 1501-12 | 1 | CABLE, 20-conductor ribbon, 12 in. long |
| 2001 | 1 | IC, 74LS00 quad NAND gate |
| 2005 | 3 | IC, 74LS04 hex inverter |
| 2010 | 1 | IC, 7413 dual 4-input NAND, Schmitt trigger |
| 2022 | 1 | IC, 7438 quad OC NAND gate |
| 2030 | 1 | IC, 74LS74 dual D flip-flop |
| 2035 | 1 | IC, 7493 4-bit binary counter |
| 2042 | 1 | IC, 74LS138 3-to-8 decoder |
| 2044 | 2 | IC, 74150 16-to-1 multiplexer |
| 2420 | 1 | IC, 4050 hex buffer, CMOS |
| 4000-44 | 1 | RESISTOR, 390 ohms, 10%, ¼W |
| 4000-49 | 1 | RESISTOR, 1 K, 10%, ¼W |
| 4000-55 | 1 | RESISTOR, 3.3 K, 10%, ¼W |
| 4000-61 | 1 | RESISTOR, 10 K, 10%, ¼W |
| 4000-66 | 1 | RESISTOR, 27 K, 10%, ¼W |
| 4000-73 | 1 | RESISTOR, 100 K, 10%, ¼W |
| 4001-97 | 1 | RESISTOR, 1 K, 5%, ¼W |
| 4001-105 | 1 | RESISTOR, 2.2 K, 5%, ¼W |
| 4001-115 | 1 | RESISTOR, 5.6 K, 5%, ¼W |
| 4001-122 | 1 | RESISTOR, 11 K, 5%, ¼W |
| 5000-16 | 1 | CAPACITOR, 220 pf, 1000 V, ceramic |
| 5010 | 1 | CAPACITOR, 0.68 uf, 20%, 15 V, tantalum |
| 5501 | 5 | CAPACITOR, .05 to 0.1 uf, 10 V min., ceramic |
| 5502 | 1 | CAPACITOR, 22 to 50 uf, 15 V min., tantalum |
| 6501 | 1 | SWITCH, miniature pushbutton, SPDT |

SEC. 3 APPLICATION NOTES REV. A

3.1 *Connections to Oscilloscope* The 472 board has three pairs of pads on the righthand side of the board, into which wire loops have been soldered. The oscilloscope probes should be connected here in order to utilize the 32-Channel Scope Display. These loops are labeled:

GRND ( )

VERT ( )

SYNC ( )

The VERT terminal is connected to the scope's Vertical Input, and the SYNC terminal to the scope's External Sync Input. A shielded cable should be used for connecting to the VERT terminal, and the shield conductor associated with the vertical input cable should be connected to the 472 GRND terminal. If a scope probe is used, then the ground lead associated with that probe should be used.

3.2 *Hints for Best Scope Picture* As a first approximation, try oscilloscope settings of about ½ V/cm, vertical sensitivity, and 10 us/cm, horizontal sweep rate. Set up for a negative sync pulse and adjust the sweep trigger level knob to visualize the 16 channels. Adjust the vertical sensitivity pot (in uncalibrated mode) to fill up the screen with the 16 channels.

Different horizontal sweep rates can be used to visualize shorter or longer instruction sequences. In order to center attention on a single point on the tracing, use the horizontal position control to move the point of interest to the center graticule line on the scope face. The delayed sweep feature (if available) is a very useful tool for focusing on an interesting instruction sequence.

3.3 *Use of Probes (Data Channels 8 & 9)* Of the 32 channels provided by the display, two are not wired into any fixed microcomputer bus lines, but are left open for the user. Two terminals are located on the left center edge of the 472 board, and may be connected via suitable leads to any interesting microcomputer signal.

Probe No. 2, leading directly to a 74150, requires TTL-level inputs - that is, 0.0 to 0.8 V logic low, 2.0 to 5.5 V logic high. Probe No. 1 leads to a CMOS buffer, with a 100 K pull-up resistor, for use in probing low-power circuitry, such as MOS memory or the CPU.

*WARNING: Care should be taken to avoid bringing these probes into contact with voltage sources outside the range of 0 to 5 volts DC, to prevent damage to the microcomputer system.*

Probe No. 2 appears as Data Channel #8, and Probe No. 1 as Data Channel No. 9, on the scope face.

SEC. 3 APPLICATION NOTES (CONTINUED) REV. A

3.4 *Repetitive Reset Mode* The 7413 reset oscillator on the 472 board repetitively requests a reset from the CPU board. This causes the computer to execute its initialization program repeatedly, starting at the reset location in program memory. Typically the pattern on the scope display will reflect permanently-stored programs, such as those in the system monitor PROM. This reset display mode is useful both in debugging hardware faults which prevent user's software from being executed, and also for educational use in understanding the microcomputer's initialization. sequence.

*NOTE:* With the 472 board installed, the 7413 reset oscillator in its socket, and the 20-conductor cable in place between the 472 and the CPU boards, the reset oscillator is in operation. In order to permit the microcomputer to execute programs at some other location, it will be necessary to defeat the reset oscillator. There are three methods of doing so:

1) Remove the 7413 temporarily from its socket on the 472 board.

2) Connect jumper wire JU1 in position on the 472 board, to the right of the 7413 socket. If two wires are soldered in, one to each side of the jumper path, they can be twisted together to defeat the oscillator, and untwisted apart to energize it.

3) Disconnect one or both sides of the 20-conductor cable between the 472 and CPU boards. However, this means of disconnection of the reset oscillator also defeats the normal operation of Channels 15 through 10 on the 32-Channel Display - namely, six selected CPU control signals - in case the display is to be used in a manner other than the repetitive reset mode.

3.5 *Repetitive Strobe Mode* Another method of providing the repetitive signal necessary for operation of the 32-Channel Display is to execute a program sequence stored in the microcomputer's memory which has two special characteristics:

1) It takes the form of a loop, i.e., at the end there is a branch instruction which causes the computer to jump back to the beginning of the program again, repeatedly executing the same sequence.

2) It includes an OUTPUT 37 instruction,normally at the beginning of the loop, which will cause the 472 board to generate a $\overline{DT37}$ strobe, serving as the needed repetitive trigger signal.

An OUTPUT 37 instruction, in a *MIKE 3* System (8080), is generated by any memory read or write instruction which references page 377 (octal) or FF (hex) in memory. In a 6800/6502 system, the appropriate memory address is page 277 (octal) or BF (hex).

Recall that it is necessary to disable the repetitive reset circuit in order to operate the display in this mode (see the preceding section).

SEC. 3 APPLICATION NOTES (CONTINUED) REV. A

3.6 *The 32-Channel Display in an 8080 System* The picture on the first page of this data sheet was shot with a Polaroid Camera attached to a Tektronix 465 Scope, connected to a Model 472 board and a *MIKE 3* microcomputer. The combined distortions and reductions of the camera and printing process make that picture more difficult to read than is the actual scope tracing. Therefore, the picture has been traced and enlarged in the figure that appears on a following page, as an illustration of the operation of the 32-Channel Display.

The program being executed takes the form of a repetitive loop. The program listing also appears on a following page. It consists of a few initial instructions which load up the H and L registers with arbitrary values;push these values onto the stack; and then load up the H, L register pair with new arbitrary numbers.

The loop portion consists of an OUTPUT 37 instruction, whose sole purpose is to create the needed strobe for the display. Then comes an XTHL instruction, which exchanges the contents of the H,L register pair, and the two top words on the general-purpose stack. Next is a second XTHL instruction, which performs the exchange again. Finally, the program jumps back to the beginning of the loop, continuing to repeat indefinitely until the computer is interrupted or reset.

The scope display is divided into upper and lower parts, labeled *DATA/CONTROL LINES* and *ADDRESS LINES*, respectively. The ADDRESS portion appears when the user depresses the Data/Address switch at the left edge of the 472 board, while the DATA portion appears when the switch is released.

The section of the display that appears in the picture begins, at the left, with the end of the OUTPUT 37 instruction. Note the 377 on the address lines.

The first XTHL instruction appears next. Referring to Data Channels 7 through 0, one can read the bits down along the vertical line: 1, 1, 1, 0, 0, 0, 1, 1. This corresponds to 343 (octal) and is the instruction code for XTHL. The 8080 is fetching this value from program memory - and if one follows the vertical line down into the ADDRESS portion of the display, one reads the associated address bits which (in octal) come out to 010 014. Checking the program listing, this is the address (in RAM as it happens) in which the demonstration program was written.

The vertical lines in the drawing were added by the illustrator - they would not appear on the scope screen itself in reality. Instead, the viewer uses the horizontal position knob on the scope to line up a point of interest beneath the center vertical line on the scope's ruled graticule.

The vertical lines were drawn so that they come in the middle of either a DBIN or WR pulse (Data Channels 13 and 12). DBIN is a signal indicating

(Continued on page 14)

# MICROCOMPUTER CODING FORM

**PROGRAM:** 32-CHANNEL DISPLAY DEMO: REPETITIVE STROBE **DATE:** 4/76

**BY:** DPM/KB **PAGE** 12 **OF** DS472 REV. B

| | LOCATION H | LOCATION L | INSTRUCTION CODE 1 | INSTRUCTION CODE 2 | INSTRUCTION CODE 3 | LABEL | OPERATION | OPERANDS AND REMARKS |
|---|---|---|---|---|---|---|---|---|
| 1 | 010 | 000 | 041 | 125 | 252 | | LXI | H,252125 LOAD UP H AND L |
| 2 | | 003 | 345 | | | | PUSH | H PUSH H,L ONTO STACK |
| 3 | | 004 | 041 | 345 | 012 | | LXI | H,012345 NEW H,L VALUES |
| 4 | | 007 | 000 | | | | NOP | NO OPERATION (FILL) |
| 5 | | 010 | 000 | | | | NOP | FILL |
| 6 | | 011 | 062 | 377 | 377 | LOOP | LDA | 377377 MEMORY WRITE - OUT 377 |
| 7 | | 014 | 343 | | | | XTHL | EXCH. H,L TOP OF STACK |
| 8 | | 015 | 343 | | | | XTHL | EXCHANGE AGAIN |
| 9 | | 016 | 303 | 011 | 010 | | JMP | LOOP LOOP AGAIN |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | | | | | | | | |
| 15 | | | | | | | | |
| 16 | | | | | | | | |
| 17 | | | | | | | | |
| 18 | | | | | | | | |
| 19 | | | | | | | | |
| 20 | | | | | | | | |
| 21 | | | | | | | | |
| 22 | | | | | | | | |
| 23 | | | | | | | | |
| 24 | | | | | | | | |
| 25 | | | | | | | | |
| 26 | | | | | | | | |

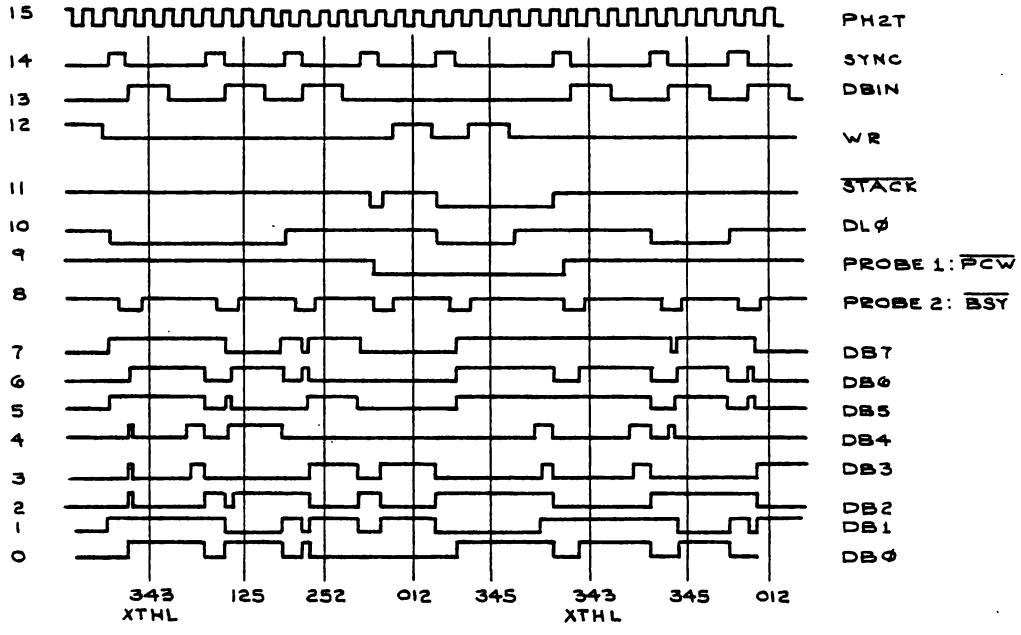
MODEL 472 DEBUG BOARD REV. B



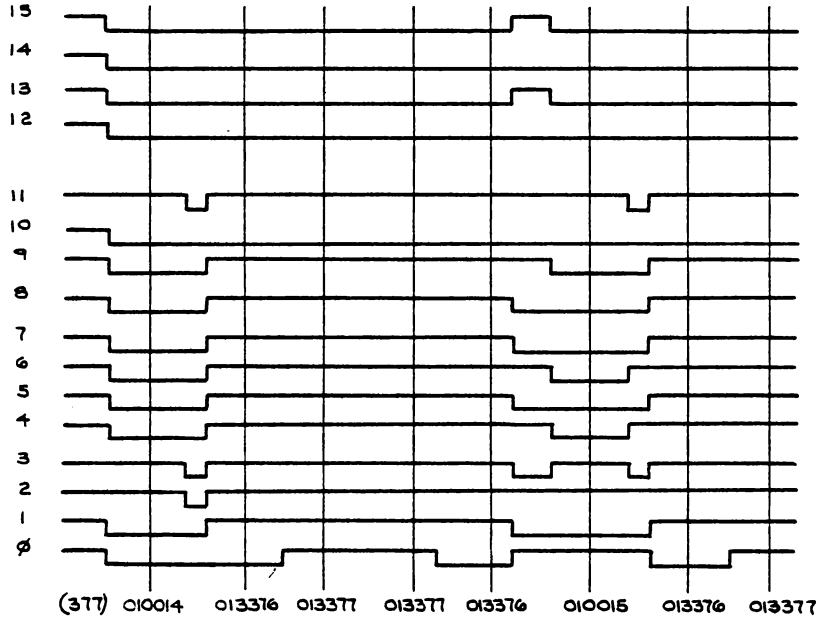
REORDER IN PADS OF 50 - CAT. NO. SP7080-$2.00 - MARTIN RESEARCH, NORTHBROOK, IL 60062

| REV | ECO | BY | CH | DATE |
|---|---|---|---|---|
| A | | | | |

## DATA/ CONTROL LINES




## ADDRESS LINES




NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY | CHECKED BY |
|---|---|
| DATE | DATE |

DRAWING NO. DS472 PAGE 13

EF4A

modular micros martin research

SEC. 3 APPLICATION NOTES (CONTINUED)

3.6 *The 32-Channel Display in an 8080 System (Continued)*

that the 8080 is inputting information on its data bus, and WR designates that data is being written (outputted) on the data bus. Thus, the first vertical line comes beneath a DBIN strobe, since this is an instruction fetch operation.

The next two vertical lines are also input operations - they represent the 8080's pulling two data values off the stack. The data value 125 comes from memory location 013 376, and the data value 252 comes from 013 377. The stack pointer was initialized by the *MIKE 3* Monitor to location 014 000 in the stack; these two memory addresses are down one and two levels in the stack.

The following two vertical lines are write operations, falling beneath two WR pulses. Here, two data values, previously in the 8080's H and L registers, are being loaded into the stack at memory locations 013 377 and 013 376. This effectively completes the first XTHL operation. The user should compare the scope drawing with the program listing to satisfy himself that the purposes of the XTHL instruction have been carried out.

The 8080 now proceeds to its next program instruction - the second XTHL instruction, stored at RAM location 010015. Its operation is similar to the first XTHL instruction. For the sake of detail, the picture has been truncated before the end of the sequence - but, in practice, the display viewer can use the horizontal controls on the scope to view much longer program sequences.

Here, briefly, are some other points of interest in the scope display example. Data Channel 15, PH2T, is the 8080 clock signal, running at 2 MHz. SYNC occurs at the beginning of each 8080 machine cycle, and the state of the data bus at that time indicates the purpose of that cycle. (For details, see the data sheet on the Model 471 CPU board.)

Probe No. 1 was connected to the signal $\overline{\text{PCW}}$, a signal which goes low during memory write cycles. Probe No. 2 was connected to $\overline{\text{BSY}}$, which goes low when a slow memory device has been accessed, in order to prolong memory read/write cycles. Note that this busy signal goes low shortly before every vertical line in the drawing. This is because each of these operations involved a reference from 2112 RAM chips on a Model 423 PROM/RAM board which incorporates a memory wait circuit, since it uses 1702A PROMs.

# 7. Memory: ROM, RAM

The *MIKE 3* develops all of the essential control signals for interfacing to a wide variety of memory devices.

The basic *MIKE 3* system comes with 512 bytes of random access memory (RAM) and 256 bytes of programmable read-only memory (PROM). Located on the Model 423 PROM/RAM board, this array can be expanded to 1K bytes of RAM, and 2K bytes of PROM.

For further memory expansion, the user can add one or more Model 405 4K RAM boards. Each Model 405 contains room for up to 4,096 bytes of memory storage.

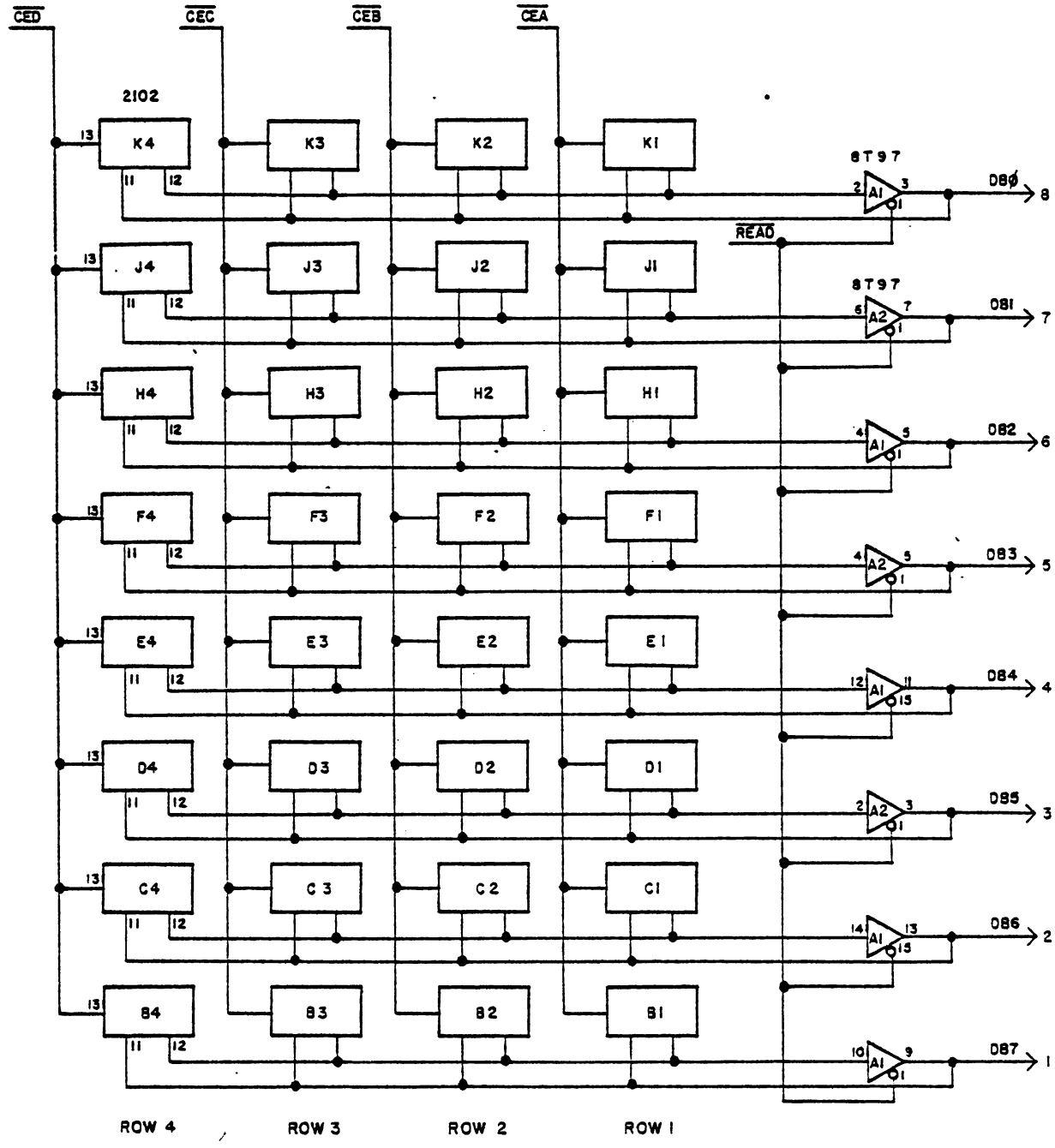
The *MIKE 3* accommodates slow memory with optional memory wait circuitry. The details are described in the data sheet for the Model 471 CPU board, Sec. 1.10, in the *CPU* section of this Manual.

Note that for optimal system speed, it is more efficient to remove the one-shot from the CPU board, and to install memory wait circuits only when necessary on the various memory boards installed in the system. Then the CPU will wait only when slow memory makes it necessary.

For the convenience of the reader, the schematics (only) for the Model 405 and Model 423 boards appear in this section of the *MIKE 3* Manual. For full circuit descriptions, see the appropriate pages of the *MIKE 2 Manual*.

# FOUR KILOBYTE RANDOM ACCESS MEMORY
## MODEL 405

| REV |
|---|
| A |
| |
| |



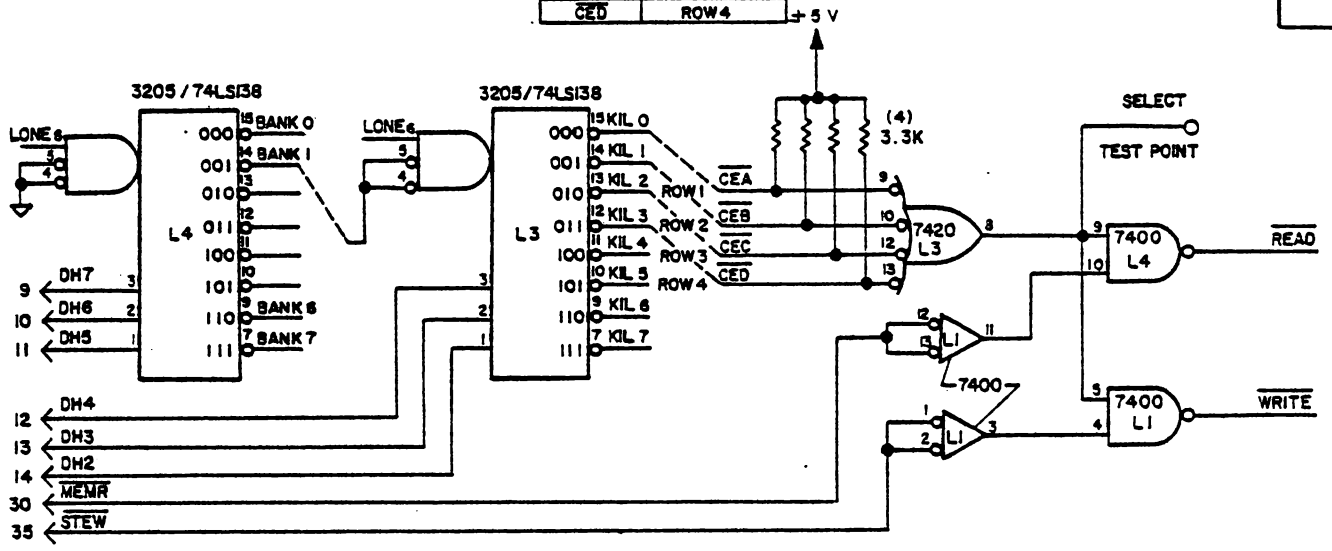

| | | |
|---|---|---|
| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY ROECK<br>DATE 11-10-75 | CHECKED BY<br>DATE |
| | | DRAWING NO.<br>DS405 (Page 4) |

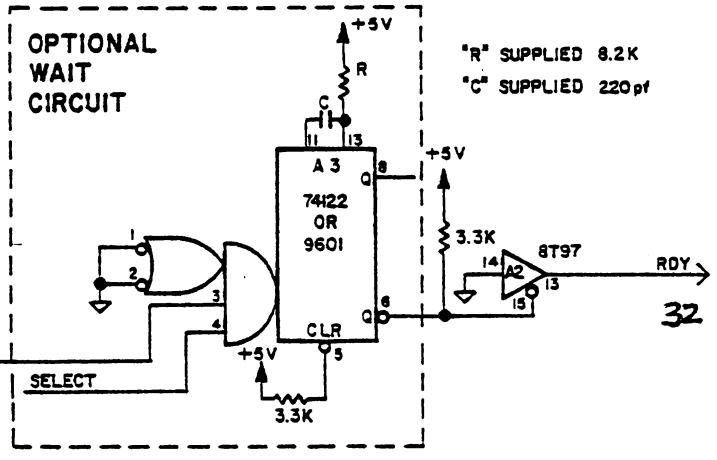
EF4A

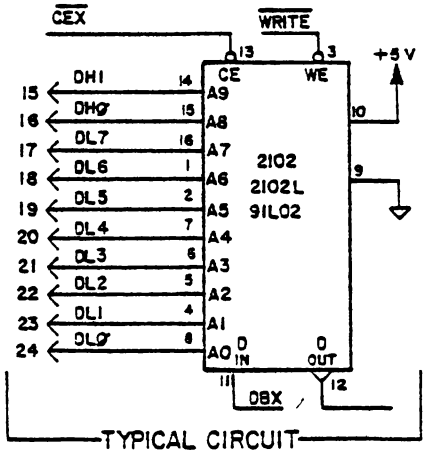
modular micros
martin research

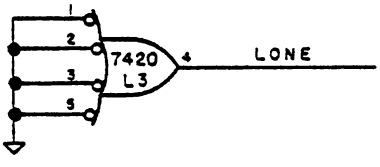
| REV |
|---|
| A |
| B |

| SIGNAL | ENABLES |
|---|---|
| $\overline{CEA}$ | ROW 1 |
| $\overline{CEB}$ | ROW 2 |
| $\overline{CEC}$ | ROW 3 |
| $\overline{CED}$ | ROW 4 |




| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY SCHNEIDER | CHECKED BY |
|---|---|---|
| | DATE 11-10-75 | DATE |

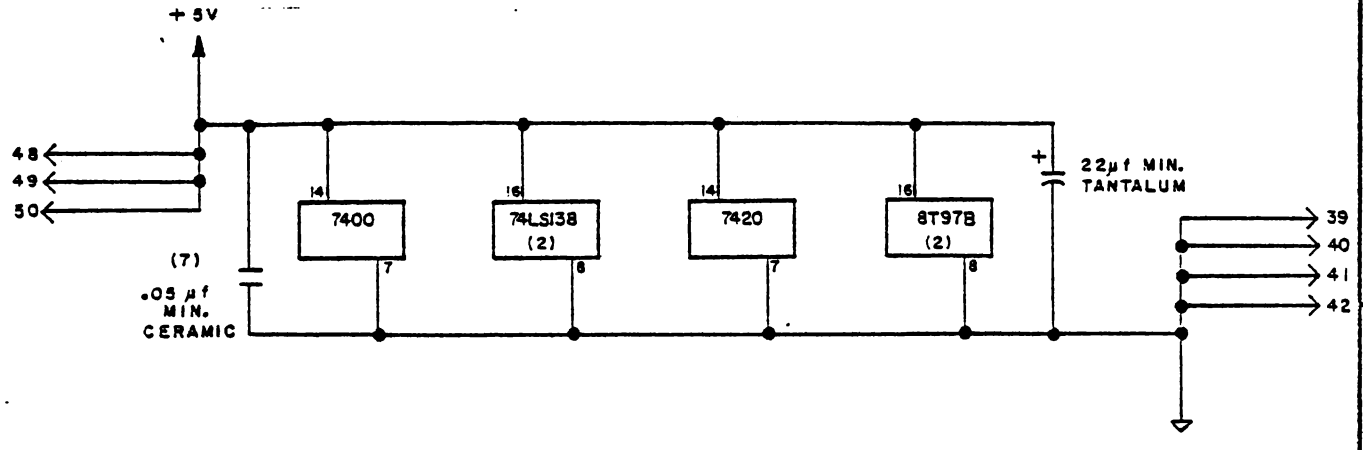
DRAWING NO. DS 405 (Page 5)

EF4A

modular micros
martin research

| REV |
| --- |
| A |
| |
| |

POWER BUSSING




NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

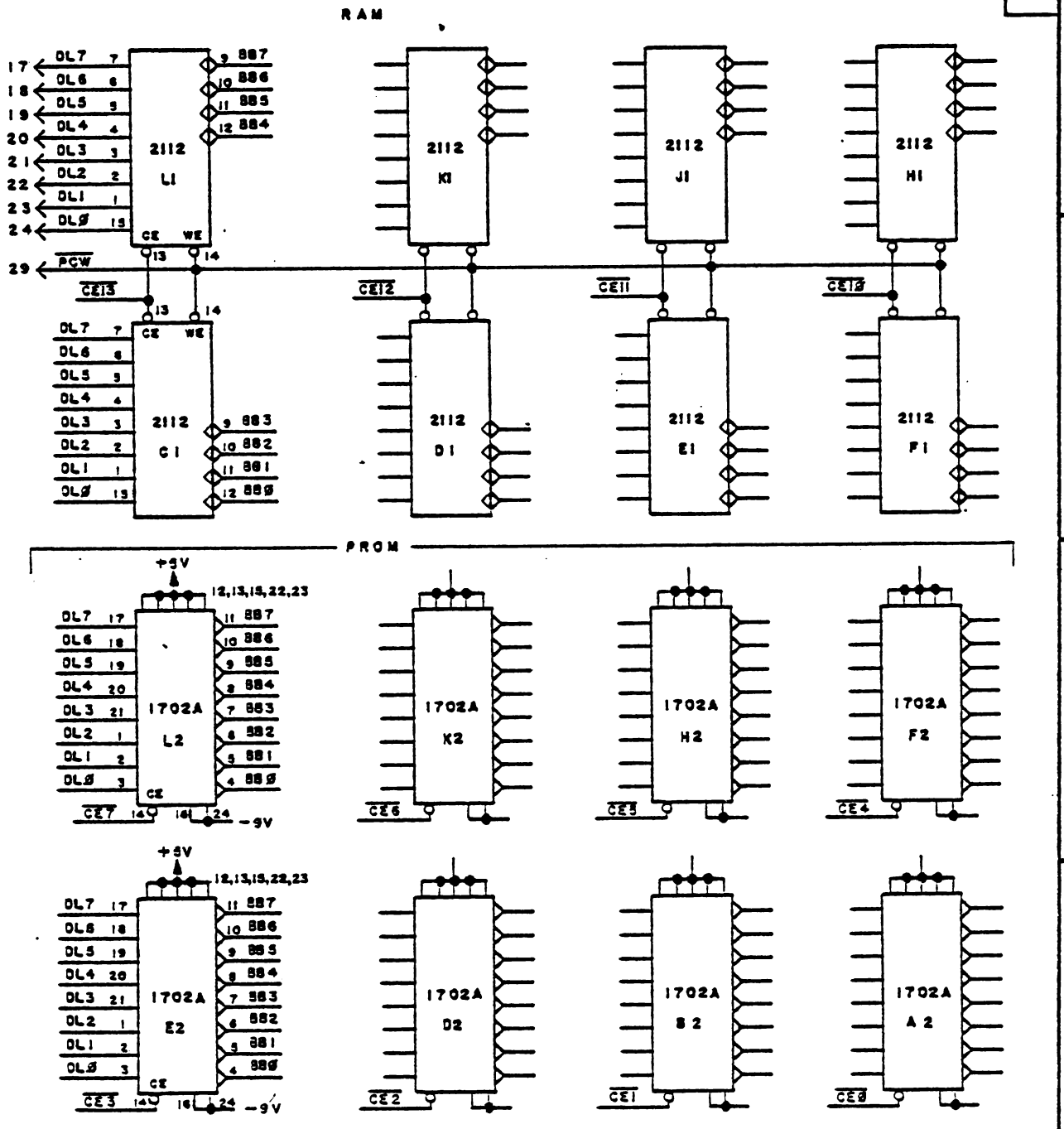
| DRAWN BY ROECK | CHECKED BY |
| --- | --- |
| DATE 11-10-75 | DATE 12/[illegible]/76 |

DRAWING NO. DS 405 (Page 6)

EF4A

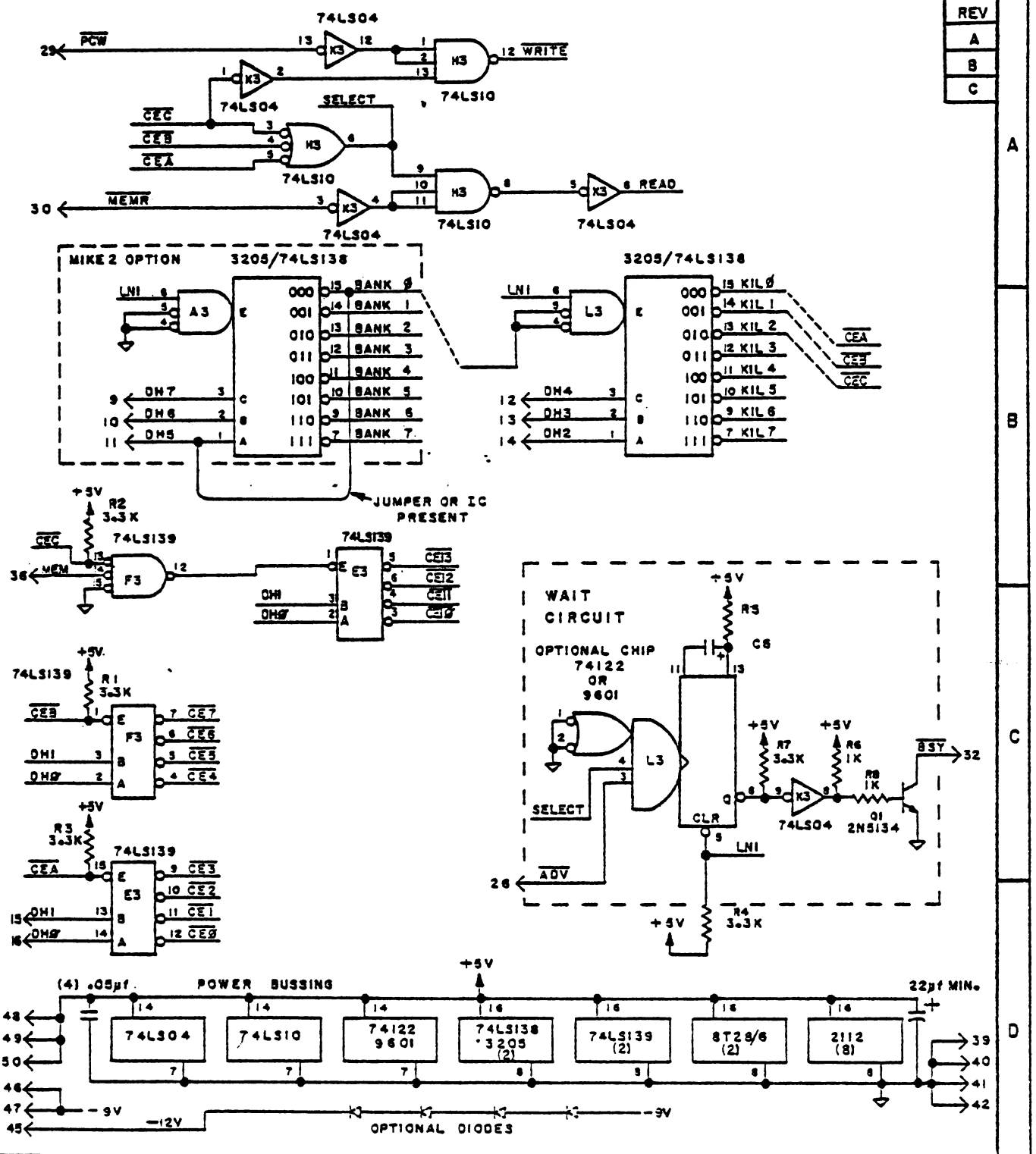
modular micros martin research

| REV |
|---|
| A |
| |
| |




| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY ROECK | CHECKED BY | DRAWING NO. DS 423 (Page 5) |
|---|---|---|---|
| | DATE 11-20-75 | DATE | |

| REV |
|---|
| A |
| B |
| C |

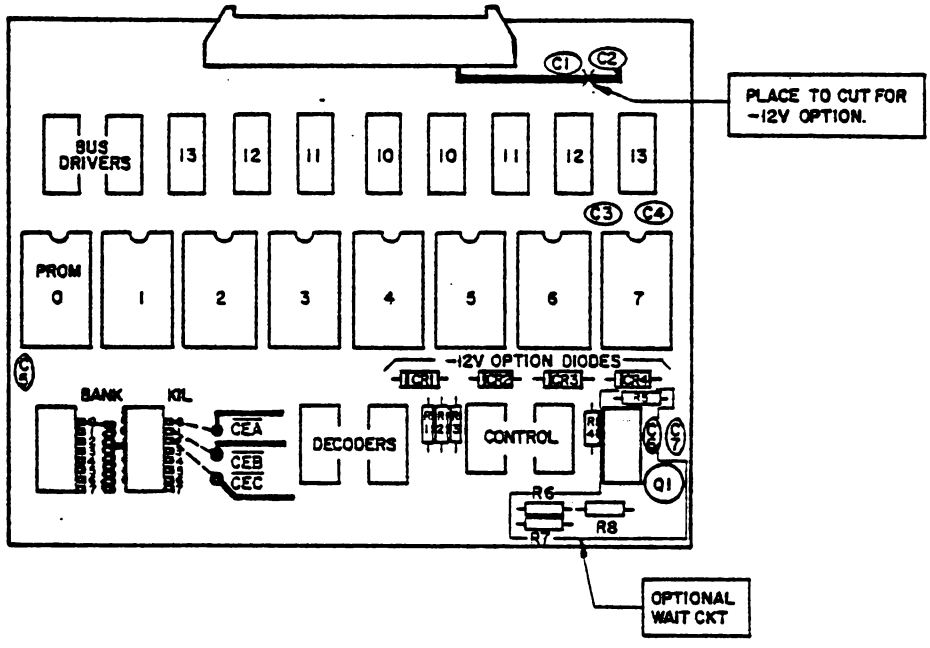



| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Roeck | CHECKED BY | DRAWING NO. DS 423 (Page 4) |
|---|---|---|---|
| | DATE 11-20-75 | DATE | |

EF4A

modular micros
martin research

| REV |
| --- |
| A |
| |
| |

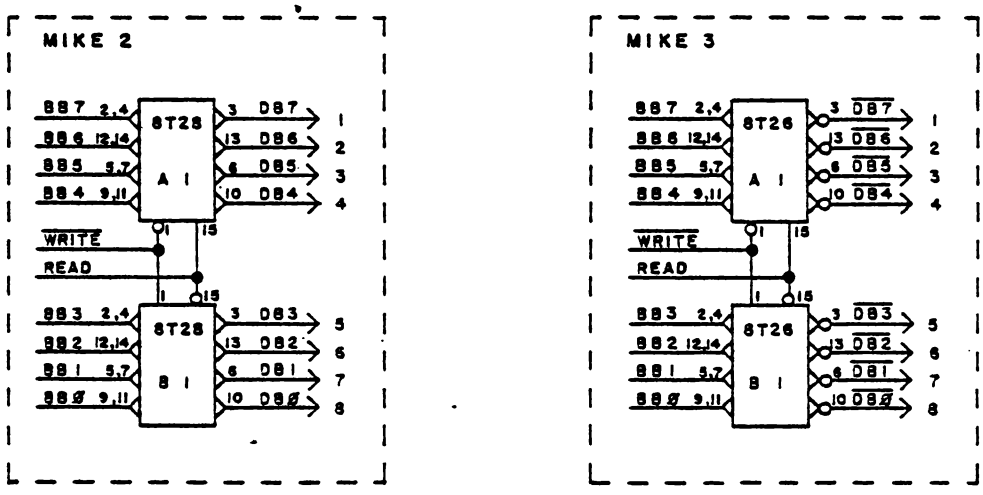



| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY DM | DRAWING NO. DS 423 (FIG 3.1.1) |
| --- | --- | --- | --- |
| | DATE 11-21-75 | DATE 11-21-75 | |

EF4A

modular micros martin research

| REV. |
|---|
| A |
| |
| |




NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY ROECK | CHECKED BY DM | DRAWING NO. DS 423 (Page 6) |
|---|---|---|
| DATE 12-9-75 | DATE 11-9-75 | |

EF4A

# 8. Input and Output Devices

REV. A

The *MIKE 3* supports three commonly-used input/output modes: the standard 8080 mode; memory-mapped I/O; and 8008-compatible I/O instructions.

The 8080 includes two special I/O instructions, INP and OUT, which are separate from instructions referencing memory. The second byte of the instruction is an eight-bit I/O port address, which is used to address up to 256 input ports and 256 output ports. Special I/O strobe signals, $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$, are developed by the CPU board and are available on the *modular micro* bus.

A second I/O mode is supported: simply defining given pages in memory as I/O ports, and using memory read and write instructions to transfer data to and from these I/O ports. The hardware associated with these ports uses $\overline{\text{MEMR}}$ and $\overline{\text{MEMW}}$ ($\overline{\text{STEW}}$) rather than $\overline{\text{IOR}}$ or $\overline{\text{IOW}}$.

A third mode, compatible with the 8008 microprocessor, is also used. A specialized universal I/O strobe signal, $\overline{\text{DIN}}$, is produced on the CPU board during instructions which reference selected portions of memory. The port address is available to output port hardware on the high-order address bus (DH7-DH0), while the output data is available on the *low-order address bus* (DL7-DL0)--rather than on the data bus, as is usual with 8080-mode I/O instructions.

This 8008-compatible output mode is used in the *MIKE 3* in scanning the keyboard and outputting to the digits on the Model 420 Console board.

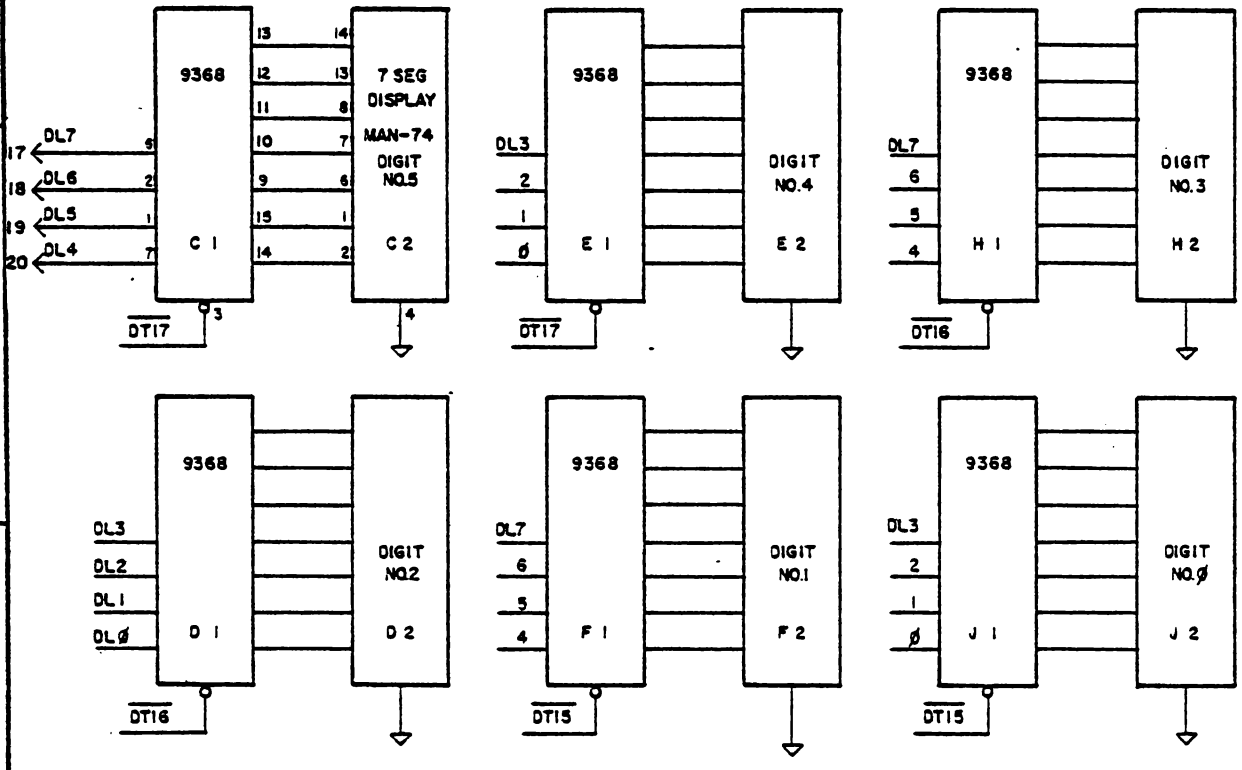
For a full explanation of these I/O modes, see the data sheet on the Model 471 Central Processing Unit in the *CPU* section of the Manual.

Note that the *MIKE 3* Monitor program--in the *Software* section of the Manual--contains subroutines for outputting to the digits and for inputting from the keyboard. These may be called by the user in his own specialized programs.

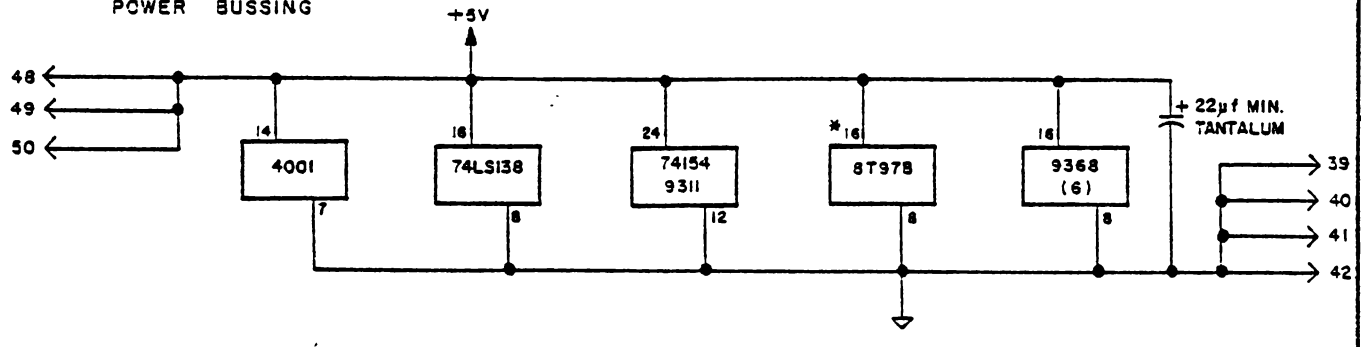
For the user's convenience, the schematics (only) of the Model 420 Console board and the Model 9001 Universal I/O Kit are included in this section. For full circuit descriptions, see the *MIKE 2 Manual*.

| REV |
|---|
| A |
| |
| |

## DISPLAY



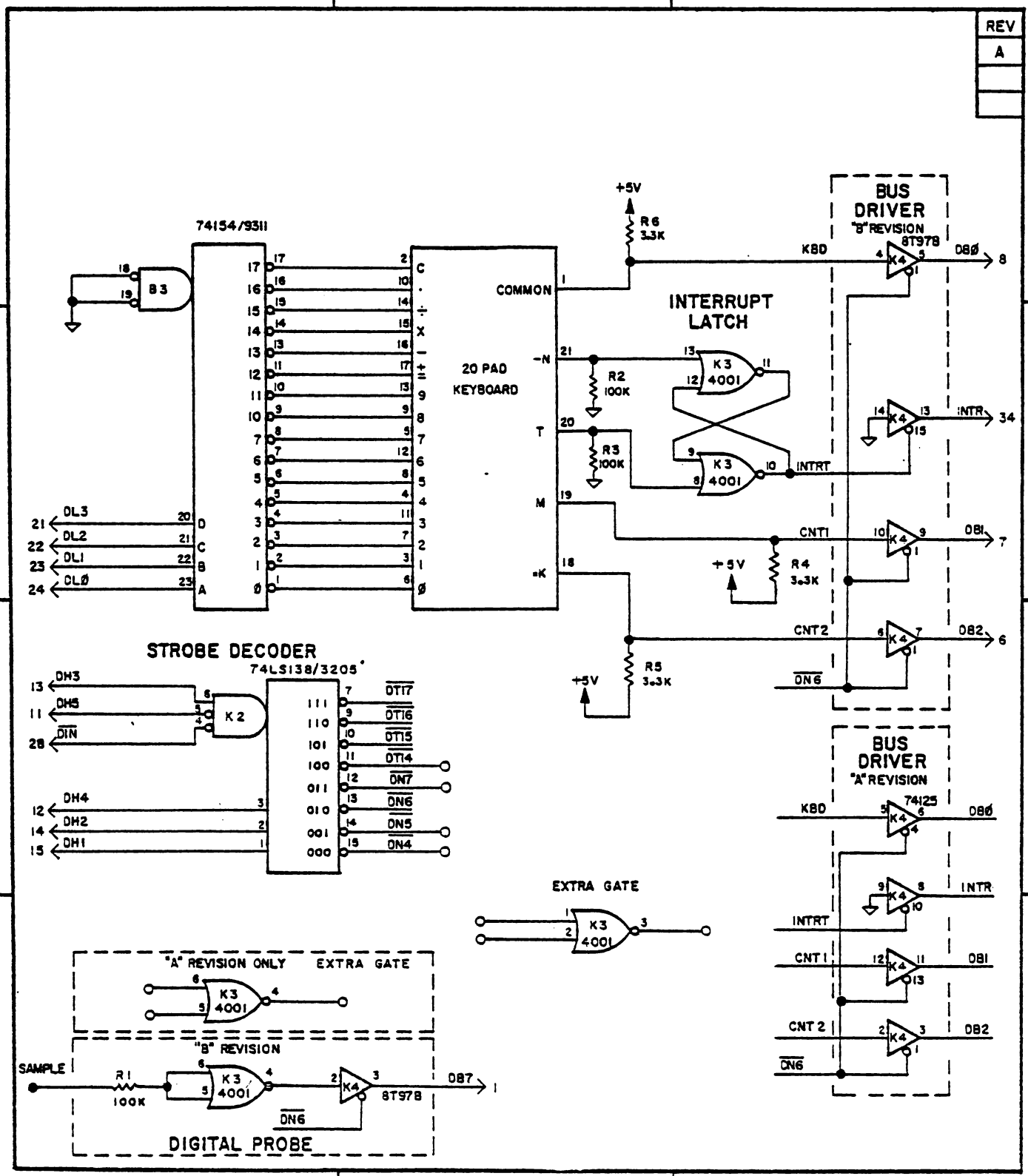

POWER BUSSING




* "A" REVISION 74125

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY D.T. | DRAWING NO. DS 420 (Page 4) |
|---|---|---|---|
| | DATE 11-19-75 | DATE 11-17-75 | |

EF4A



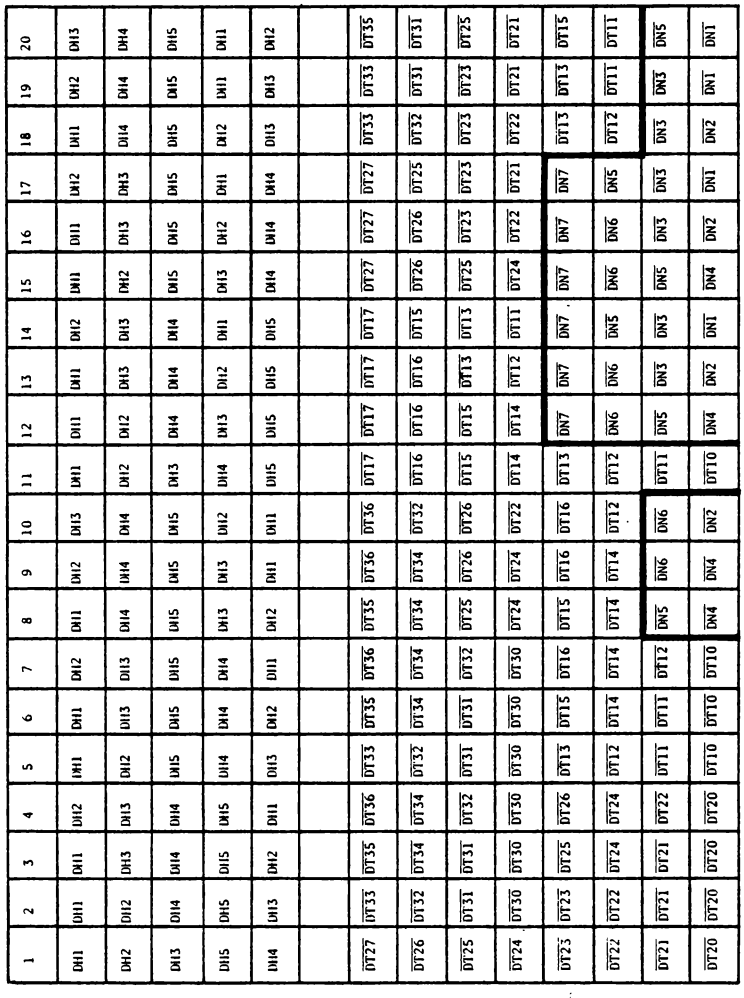

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY Schneider | CHECKED BY DTL | DRAWING NO. |
|---|---|---|
| DATE 11-19-75 | DATE 11-19-75 | DS 420 (PAGE 5) |

EF4A

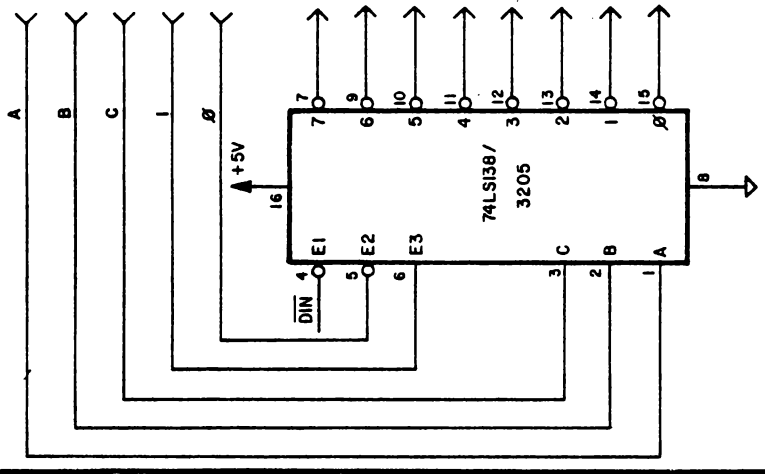
modular micros
martin research

| REV |
|---|
| A |
| |
| |



NOTE: PIN CONNECTIONS ON MODULAR MICRO BUS:

| | | | |
|---|---|---|---|
| $\overline{DIN}$ | 28 | DH3 | 13 |
| DH5 | 11 | DH2 | 14 |
| DH4 | 12 | DH1 | 15 |




NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

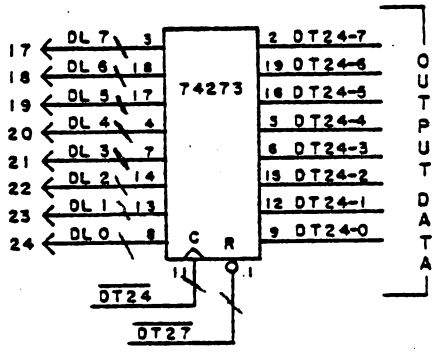
| | |
|---|---|
| DRAWN BY Schneider | CHECKED BY KB |
| DATE 11/29/79 | DATE 12/1/79 |

DRAWING NO. DS 9001 FIG 1.4.1

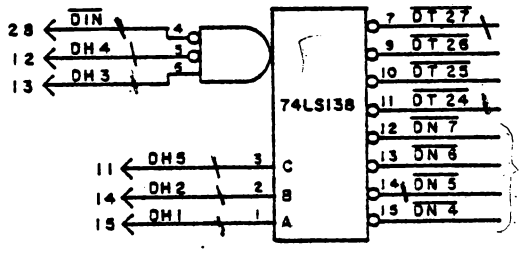
EF4A

| REV |
|---|
| A |
| |
| |

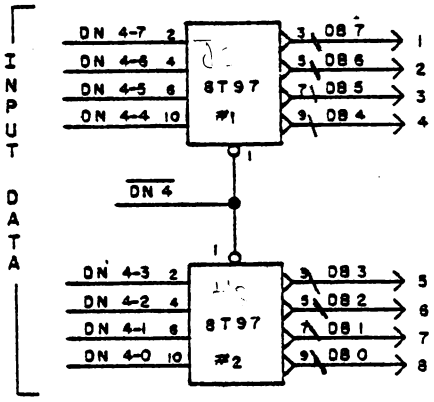
## OUTPUT PORT Ø24



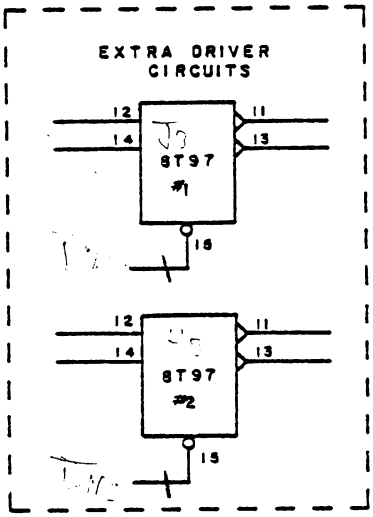

## I/O STROBE DECODER



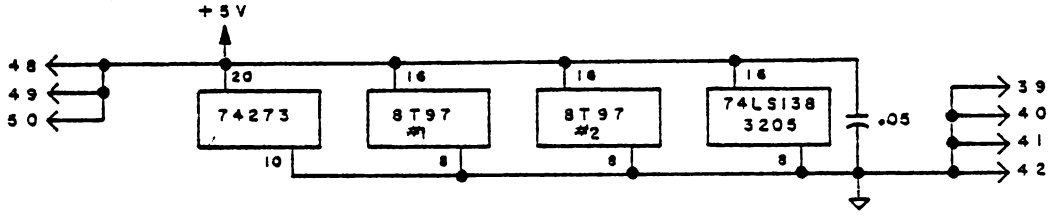

## INPUT PORT ØØ4




## EXTRA DRIVER CIRCUITS




## POWER BUSSING




| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY Schneider | CHECKED BY DH | DRAWING NO. DS 9001-1 (Page 7) |
|---|---|---|---|
| | DATE 11-19-75 | DATE 11-19-75 | |

EF4A

modular micros martin research

# ICs interface keyboard to microprocessor

by Donald P. Martin and Kerry S. Berland
*Martin Research Ltd., Chicago, Ill.*

A compact, economical interface between a keyboard and a microprocessor can be designed with only three integrated-circuit chips. The ICs are a 5740 MOS scanning keyboard encode, a 2812 MOS first-in/first-out (FIFO) memory, and a 74125 quad three-state buffer. All three can be mounted with the standard array of keyswitches and diodes on a single circuit board.

The 5740 keyboard encoder has 10 scan inputs and nine scan outputs. A unique combination of one input and one output is assigned to each key, adding up to 90 keys in all. The keys are wired between the scan inputs and the outputs with a diode in series, as shown in the circuit diagram. The diodes block sneak signal paths and eliminate "phantom key" effects if several keys are pressed at the same instant.

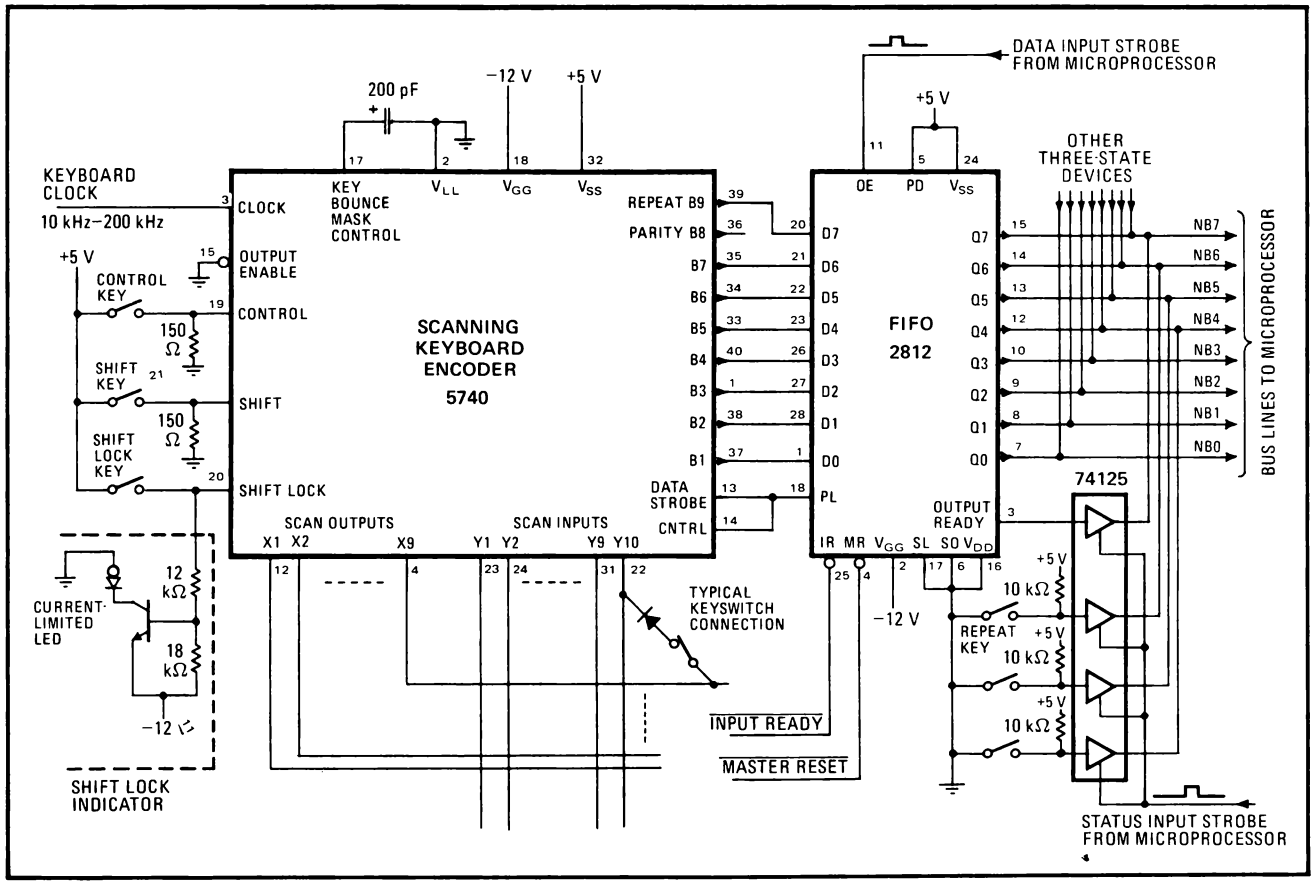
Internal ring counters simultaneously scan both the key matrix and an internal read-only memory. When a key is pressed, the ROM word corresponding to that key is transferred into a one-character nine-bit output latch. The word includes the seven-bit ASCII code for the character, parity bit B8 (which is not used in this design), and the selective repeat bit B9. The 5740 requires an external clock oscillator in the 10-to-200-kilohertz range to drive the scanning counters; this can be obtained from the main timing circuit of the microprocessor.

The internal circuitry of the encoder also performs other necessary functions; it suppresses keybounce effects, responds to key closures even if the previous key has not yet been released, and senses the shift, shift lock and control mode keys.

When the encoder recognizes a keystroke, it sets its data strobe output high. This terminal is wired directly to the encoder's data strobe control, an input terminal that resets the encoder output on the next falling edge of the keyboard clock. The data word is thus available at the encoder for only one clock period.

If no more data storage were provided than the one character stored by the encoder, the microprocessor would have to test for new keyboard data at a rapid rate. This requirement would be a severe constraint on the software, so the 2812 FIFO is included in the interface to provide storage of up to 32 characters.

The keystroke that sets the encoder data strobe high also strobes the parallel load (PL) input of the FIFO and loads the ASCII character into the FIFO. The loaded character moves down through the 32 positions until it either reaches the output or is stopped by a previously




**Interface.** Three ICs connect keyboard to microprocessor. Encoder provides up to 90 keys. FIFO stores 32 characters. Connections to microprocessor are made through three-state devices to the same 8 bus lines as used by other data sources. All three ICs can be mounted with keyswitches and diodes on a single circuit board, so when keyboard is not included in system, neither are interface components.

stored character. When there is at least one character stored in the FIFO, the output ready signal goes high. This signal is periodically tested by the microprocessor to see whether there is new data from the keyboard. With the FIFO providing buffer storage, the microprocessor needs to test for input data far less frequently.

The FIFO's parallel dump (PD) control is permanently enabled (wired to +5 volts). However, the parallel dump function is also internally gated with the output enable (OE) terminal; therefore the first-received character will not be dumped until the OE terminal is activated. Thus a single strobe to the 2812 first reads the keyboard word into the microcomputer, then dumps the word out of the FIFO, moving the next keyboard character into the output position. The delay between the leading edge of the data input strobe and the appearance of valid data on the microprocessor input bus is less than 400 nanoseconds for the 2812.

When power is first applied, the FIFO registers are cleared by a signal from the master reset circuit of the microprocessor. This signal goes low for a fraction of a second, preventing the FIFO from taking on initial random states that could be interpreted as keyboard data.

To load information into the microprocessor, status input and data input instructions are used. The microprocessor periodically pulses the status input strobe line. This pulse activates the 74125 three-state buffer, which puts the FIFO's output ready bit on the high-order input bus line of the microprocessor. (This is input bit 7, or NB7.) The microprocessor tests this bit to see whether keyboard data is available; if the bit is high, indicating that a character is stored in the FIFO, the microprocessor executes a data input instruction. This instruction activates the output enable terminal of the FIFO, and impresses the keyboard data word on to the input bus to the microprocessor.

The seven lower-order bits of the keyboard data word are the ASCII-encoded character. The high-order bit, NB7, is the selective repeat bit B9. The repeat switch is connected to the next-highest bit through an extra three-state buffer. The repeat function is implemented easily through a few instructions stored in the ROM. □

Reprinted from **Electronics**, March 6, 1975

Copyright 1975 by McGraw-Hill Inc. 1221 Avenue of the Americas, New York, N.Y. 10020

REV A

```
                                ;
                                ;       HERE TO APPEAR TO PAUSE TO OPERATOR -->
                                ;       UNLESS A KEY IS DEPRESSED
                                ;
                                ;       CONDITION CODE SIGN = 1 IF NO KEYS DEPRESSED
                                ;       CONDITION CODE SIGN = 0 IF KEYS DEPRESSED
                                ;       REGISTER C CONTAINS DEPRESSED KEY NUMBER
                                ;       DESTROYS REGISTERS A,B,C
                                ;
                                ;       NOTE:   KEYS ARE NOT DEBOUNCED (USE RST030 FOR DEBOUNCED READ)
                                ;
000040  325                     RST040  PUSH    D               . SAVE D,E REGISTERS
000041  036                             MVI     E,0200          . OUTPSIDE LOOP COUNTER
000042  200
                                ;
                                ;       INNER LOOP OF A FEW MILLESECONDS (10 MS.)
                                ;
000043  035                     RST043  DCR     E               . ANY MORE TIMES THRU THE LOOP ?
000044  372                             JM      RST056          . NOPE --> POP D, E RETURN
000045  056
000046  000
000047  006                             MVI     B,H(C8)-1       . MSB OF TIME CONSTANT OF A MS (SO IT FITS)
000050  002
000051  327                             RST     2               . WAIT 10 MS.
000052  377                             RST     7               . ANY KEYS DEPRESSED ?
000053  372                             JM      RST043          . NOPE, GO ON LIKE THIS FOR 1.0 SECONDS
000054  043
000055  000
000056  321                     RST056  POP     D               . RESTORE D AND E REGISTER
000057  311                             RET                     . RETURN
                                ;
                                ;       HERE TO DEBOUNCE KEYS
                                ;       DESTROYS REGISTERS A,B,C
                                ;
000060  006                     RST060  MVI     B,H(C100)-1     . MSB OF TIME CONSTANT  (SO IT FITS)
000061  050
000062  327                             RST     2               . WAIT 100 MS BEFORE TESTING KEYS AGAIN
000063  377                             RST     7               . ANY KEYS DEPRESSED ?
000064  362                             JP      RST060          . LOOP UNTIL NO KEYS DEPRESSED
000065  060
000066  000
000067  311                             RET                     . RETURN


                                ;
                                ;       HERE TO SEE IF ANY KEYS DEPRESSED ON KEYBOARD
                                ;       CONDITION CODE SIGN = 1 NO KEYS DEPRESSED
                                ;       CONDITION CODE SIGN = 0 KEYS DEPRESSED
                                ;       REGISTER C CONTAINS KEY NUMBER
                                ;       DESTROYS REGISTER A,B,C
                                ;
000070  001                     RST070  LXI     B,INP6*256+020  . INP 006 INSTRUCT, NUMBER OF KEYS TO POLL
000071  020
000072  315
                                ;
                                ;       LOOP TO CHECK NEXT KEY (MUST POLL EACH OF 16 KEYS)
                                ;
000073  015                     RST073  DCR     C               . DECREMENT KEY LOOP COUNTER
000074  370                             RM                      . NO MORE KEYS TO POLL, RETURN SIGN=1
000075  012                             LDAX    B               . READ FROM KEYBOARD TO SEE IF KEY DEPRESSED
000076  017                             RRC                     . ISOLATE LOW ORDER BIT  (KEY DEPRESSED FLAG)
000077  322                             JNC     RST073          . THIS KEY NOT DEPRESSED, CHECK NEXT KEY
000100  073
000101  000
                                ;
                                ;       KEY IS DEPRESSED, NOW HANDLE SHIFT KEYS (FOUR FUNCTIONS POSSIBLE)
                                ;
                                ;       VALUE   FUNCTION        SHIFT KEYS DEPRESSED
                                ;       -----   --------        --------------------
                                ;
                                ;       00-15   F0              NO SHIFT KEYS
                                ;       16-31   F1              M SHIFT KEY
                                ;       32-47   F2              =K SHIFT KEY
                                ;       48-63   F3              M AND =K SHIFT KEYS (BOTH)
                                ;
                                ;       NOTE:   REG A = 011111KM (WHERE KM ARE THE SHIFT KEYS)
                                ;
000102  346                             ANI     003             . ISOLATE THE KM SHIFT KEYS
000103  003
000104  007                             RLC                     . MULTIPLY THE
000105  007                             RLC                     . SHIFT KEY FUNCTIONS
000106  007                             RLC                     . F0,F1,F2,F3 BY 16
000107  007                             RLC                     . BY SHIFTING LEFT 4 BITS
000110  201                             ADD     C               . ADD IN KEY  (NOW SIGN =0)
000111  117                             MOV     C,A             . COPY FOR OTHERS
000112  311                             RET                     . RETURN
```

| | | |
|---|---|---|
| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R / DATE 12-10-75 / CHECKED BY DM / DATE 12-10-75 | DRAWING NO. SP2220-16 (PAGE3) |

EF4A

modular micros martin research

# 9. Power Supply

# 10. Accessories and Peripherals

REV. A

The *MIKE 3* requires external power supplies providing voltages of +5, +12, and -12, regulated ±5%.

(The regulation on the negative supply, -12, may be relaxed when it is being used in a minimal system only to provide the 8080 with -5 V and the PROM with -9 V. The 8080 -5 V requirement is regulated on the CPU card with a zener diode, and the PROM voltage requirement is not stringent. A negative supply of -11.0 to -12.6 volts is adequate.)

The currents required depend on the number of boards added to the system, and on the number of optional memory chips that are inserted into the memory boards. The minimal *MIKE 3* system--with a Model 471 (431) CPU board; a Model 420 Console board; and a Model 423 PROM/RAM board with 256 bytes of PROM and 512 bytes of RAM--needs +5 V at 2.0 A max.; 12 V at 75 ma max.; and -12 V at 45 ma max.

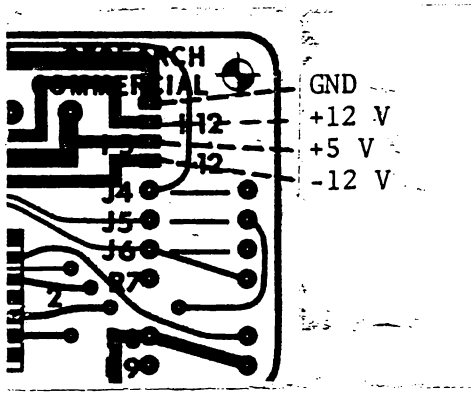
The *modular micro* bus dedicates 12 lines to power supply connections:

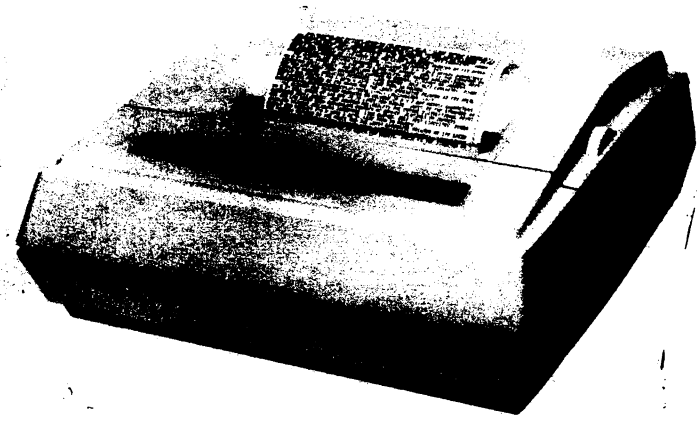
| | |
|---|---|
| 39-42 | SYSTEM GROUND |
| 43-44 | +12 VOLTS |
| 45 | -12 VOLTS |
| 46-47 | -9 VOLTS |
| 48-50 | +5 VOLTS |

The -9 volt line is usually unused in a *MIKE 3* computer. (The 423 data sheet demonstrates how the -9 V requirement of the PROMs is derived from the -12 V line.)

Typically the user connects four wires to his power supply--+12 V, +5, GROUND, and -12--and connects them to the computer at either of two places. The first site is on the Model 425 breadboard, if one is included; the correct locations for these wires are shown on the silkscreened legend on the 425 board.

The other possible site is on the Model 471 CPU board, near the rear right-hand corner of the board. To the right appears an illustration of this section of the 471 board.





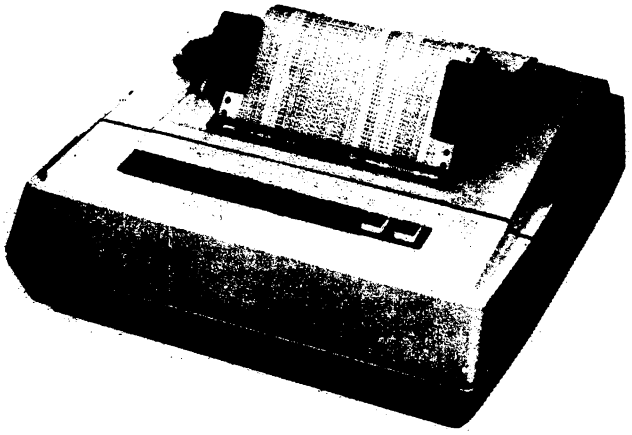
# 110 CPS PRINTER

## Features:

- **Low price impact printer**
- **110 characters per second/65 lpm**
- **80 columns of 5 x 7 dot matrix characters**
- **Designed to work with CRT's**
- **Lightweight/Tabletop design**

The **PR11000** is a desktop line printer that produces 80 columns of 5 x 7 dot matrix characters at 110 characters per second or 65 lines per minute. The impact head prints bidirectionally on 8½" roll paper using a conventional teletype ribbon. Four-ply fanfold paper up to 9½" wide may be used with optional tractor feed.

Maximum reliability is provided by a mechanism which contains no brakes, clutches, dampers or stepper motors. All control electronics including a one-line buffer and self-test circuitry are contained on a single 5" x 15" printed circuit card. The **PR11000** was expressly designed for the simplicity, reliability and extremely low cost required by current small-scale data handling systems and terminals.



Vibration and wear are minimized because the print head moves uniformly in both directions and pauses only at the end of each line. Opto-electronic sensing is used to accurately position each dot and permit characters to be printed on the fly.

A linear solenoid driving a detented ratchet and pawl reduces dead time by providing a paper feed slew rate of 600 lines per minute.

A fully packaged printer weighs less than 33 pounds, and measures 18" wide x 8" high x 22" deep. A universal input transformer allows for use worldwide.

The printer is available in several configurations from a basic mechanism to a fully packaged printer with or without tractor feed. Parallel and serial interfaces are available.

**martin research**
3336 Commercial Avenue
Northbrook, IL. 60062
(312) 498-5060

modular micros ⇄ martin research

# 11. Maintenance, Troubleshooting, Service, and Warranty Information

# 12. Resident Software and User's Programs

# 110 CHARACTER-PER-SECOND PRINTER
# MODEL PR11000

## Specifications (Standard)

### PRINT CHARACTERISTICS

| | |
|---|---|
| **Print Method** | Impact Dot Matrix |
| **Dot Size** | 0.015" |
| **Character Height** | 0.105" 7 vertical pins on 10° slant |
| **Character Width** | 0.075" 5 x 7 Matrix |
| **Character Spacing** | 10 characters/inch |
| **Line Width** | 80 characters |
| **Font** | 64 character ASCII subset standard; see optional features |

### PRINT MECHANISM

| | |
|---|---|
| **Head Motion** | Print head moves uniformly in both directions. Individual characters are printed on the fly. |
| **Print Speed** | 110 characters/second |
| **Throughput** | 65 lines/minute Bi-directional Printing |

### PAPER MECHANISM

| | |
|---|---|
| **Media** | 8½" roll paper up to 5" in diameter. 9½" wide 4-ply fanfold paper optional |
| **Line Spacing** | 6 lines/inch |
| **Feed Rate** | 100 millisecond single space 600 line/minute Slew |

### RIBBON MECHANISM

| | |
|---|---|
| **Media** | Standard ½" wide nylon ribbon on 2" spools |
| **Motion** | Self-reversing ratchet and pawl activated by print drive sprocket |
| **Extended Life** | Printing occurs only through upper half of ribbon. Spools may be manually inverted to double ribbon life. |

### INPUT POWER

| | |
|---|---|
| **Voltage** | 100, 110, 117, 235 VAC ±10%, 50 or 60 Hz selected at input transformer taps. |
| **Power** | 350 watts maximum |

### AUXILIARY POWER

| | |
|---|---|
| **Voltages** | Using the standard parallel buffered interface electronics, +5VDC @ 1 Amp and −12VDC @ 0.050 Amp are available for user interfacing |

### INTERFACING

| | |
|---|---|
| **Parallel Buffered** | Printer contains all control electronics, character generator, 80-character buffer, panel switches and self-test circuitry. |
| **Mechanism** | Inputs are print motor power, print and paper feed solenoids. Outputs are head position and end-of-line sensors. |

| | |
|---|---|
| **SIZE** | 18" wide x 8" high x 22" deep |
| **WEIGHT** | Less than 33 lbs. |

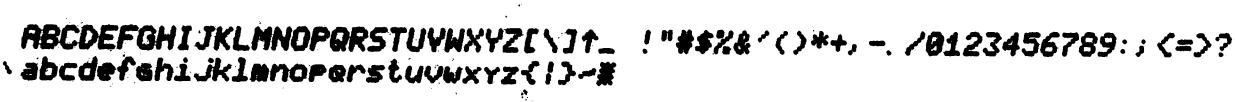
### TEMPERATURE & HUMIDITY

| | |
|---|---|
| **Operating** | 40° to 105°F, 20 to 90% |
| **Non Operating** | −30° to 160°F, 5 to 95% |

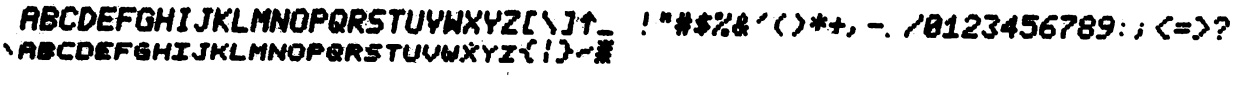
### OPTIONAL FEATURES

| | |
|---|---|
| **Tractor Feed** | Permits feeding of carbon interleaved fanfold forms up to four plies. |
| **RS232 Interface** | Special: each Model PR11000 comes with an assembled and tested Model 437 board, providing an EIA-style interface. Fitting inside the PR11000 cabinet, and included at no extra charge, the Model 437 allows connection of the printer to serial interface equipment. Suggested matching board: a UART-equipped Model 424 Serial Interface board. |
| **Upper/Lower Case** | Standard: upper case and small upper case (see second sample line below). Other lower case fonts are optional. |

| | |
|---|---|
| **U.S. price** | **$1,600.00, FOB Northbrook** |
| **Delivery** | **30 days ARO** |




Print sample of upper case and standard lower case




Print sample of upper case and small upper case

**martin research**

**3336 commercial avenue / northbrook, illinois 60062 / (312) 498–5060**

## WARRANTY POLICY

*Martin Research,* Manufacturer, warrants that the products delivered to the purchaser are free from defects in material and workmanship under normal use and service. Manufacturer's obligation under this warranty is limited to replacing or repairing or giving credit for, at its option, at its factory, any product which shall, within one (1) year after date of invoice, be returned to the Manufacturer's factory of origin, transportation charges prepaid, and which are, after examination, disclosed to the Manufacturer's satisfaction to be thus defective. This Warranty is expressed in lieu of all other warranties, expressed, statutory, or implied, including the implied warranty of fitness for a particular purpose, and of all other obligations or liabilities on the Manufacturer's part, and it neither assumes nor authorizes any other person to assume for the Manufacturer any other liabilities in connection with the sale of said articles.

This Warranty shall not apply to any of such products which shall have been repaired or altered, except by the Manufacturer, or which shall have been subjected to misuse, negligence, or accident. The aforementioned provisions do not extend the original warranty periods of any product which has either been repaired or replaced by the Manufacturer.

## RETURN POLICY

### I. *AUTHORIZATION REQUIRED*

Before returning any equipment to the Manufacturer for repair, replacement, or refund, authorization must first be received from Martin Research. Written authorization is normally required.

### II. *SHIPPING, PACKING, AND INSURANCE*

Equipment returned to the Manufacturer should be securely packed in a rigid container with ample cushioning material. Due precautions must be taken with electrically sensitive devices, such as MOS (metal oxide semiconductor) circuitry, including packing in conductive material. *Equipment should never be shipped unless it is insured for its full value.* In case of damage, the party shipping the equipment will be notified promptly. As a matter of law, and of policy on the part of shipping agencies, the responsibility for filing damage claims rests with the shipping party, though Martin Research will verify any such claim at the shipper's request. Since payment by shipping agencies on claims of damage may not be prompt, we urge our customers to take extra care in packaging equipment for shipment.

### III. *DAMAGED AND OUT-OF-WARRANTY EQUIPMENT*

If it appears to the Manufacturer that returned equipment has been subjected to misuse, negligence, or accident by the user, the user shall be notified promptly. The Manufacturer retains the option to return such equipment unrepaired, and to charge for shipping in such event. If the user has so authorized, the Manufacturer will repair such equipment, or equipment whose warranty period has expired, for any replacement parts deemed necessary at the rates shown on its current Replacement Parts List, and charging for service at the current hourly rate for electronic repair. Estimates will be provided upon request.

The *MIKE 3 DEMO PROM* is a single 256-byte PROM which adds a number of useful functions to a *MIKE 3* microcomputer. These include a digital clock program (using the six LED digits on the Model 420 Console board for the display); a program for converting hexadecimal values to decimal; and routines for testing the hex display, the keyboard, and a block of RAM starting at any location.

The *DEMO PROM* is specifically designed for operation in a *MIKE 3* system. It requires the *MIKE 3 Monitor* program for some of its subroutines, and uses the Model 420 digits for display purposes.

Usually the *DEMO PROM* is inserted as the second PROM on the Model 423 PROM/RAM board. Then the keyboard is used to cause the computer to jump to the desired subroutines in the *DEMO PROM*.

With the *octal* version of the *MIKE 3 Monitor* installed in the computer, simply typing 9 causing the computer to jump to location 001 000, giving control to the *DEMO PROM*.

With the *hexadecimal* monitor, the M shift key must be depressed at the same time as the 9 key is pressed, to give control to the *DEMO PROM*.

Once the *DEMO PROM* has control, other keys may be pressed to start the various routines supported by the PROM, as follows:

| KEY | FUNCTION |
|---|---|
| 0 | Return to Monitor. |
| 1 | Check 420 display digits. Digits count 111111; 222222; 333333; 444444;.....999999; AAAAAA; bbbbbb; CCCCCC; dddddd; EEEEEE; FFFFFF; 111111; 222222 (etc.) changing once every 1.3 second. |
| 2 | Test keyboard. When a key is depressed, its value (in hex) appears on all six digits on the Console board. |
| 3 | Test RAM. The address for the first RAM location is first set up in the H and L registers, using the LOAD H and LOAD L keys (under control of the Monitor program). Then control is given to the *DEMO PROM*, and 3 is pressed. The program tests each RAM location, and the first address at which RAM fails (or where no RAM exists) appears on the display. |
| 4 | Hex to decimal. A hexadecimal value entered on the keyboard is translated to decimal on the display. |
| 5 | 24-hour digital clock. The keyboard is used to enter hours, minutes, and seconds (in ordinary decimal format). |

MIKE 3 DEMO PROM

| REV | ECO | BY | CH | DATE |
|---|---|---|---|---|
| B | | | | |
| | | | | |
| | | | | |

LISTING IN OCTAL CODE OF *MIKE 3* DEMO PROM

```
000/ 337 171 376 010 322 000 001 006
010/ 001 306 021 117 012 117 305 367
020/ 311 031 032 071 100 323 157 364
030/ 377 307 036 000 173 346 017 117
040/ 315 050 001 347 034 303 034 001
050/ 171 207 207 207 207 201 006 337
060/ 117 012 006 335 012 006 333 012
070/ 311 337 315 050 001 302 071 001
100/ 053 371 026 002 041 000 000 071
110/ 035 302 110 001 043 006 337 114
120/ 012 006 335 115 012 006 333 174
130/ 205 117 025 312 137 001 167 024
140/ 226 012 312 110 001 025 302 104
150/ 001 061 000 014 303 125 000 315
160/ 310 001 137 315 056 001 315 310
170/ 001 147 117 315 062 001 315 310
200/ 001 157 117 026 346 016 304 015
210/ 100 100 100 302 207 001 025 302
220/ 205 001 026 360 025 302 224 001
230/ 026 001 175 016 140 315 275 001
240/ 157 006 333 115 012 174 016 140
250/ 315 275 001 147 006 335 114 012
260/ 173 016 044 315 275 001 137 026
270/ 337 032 303 293 001 292 047 107
300/ 221 076 000 237 127 240 024 311
310/ 337 171 207 207 207 207 365 337
320/ 361 261 311 337 175 207 207 207
330/ 207 261 157 107 257 117 306 001
340/ 047 322 345 001 014 005 302 336
350/ 001 006 335 026 333 137 046 337
360/ 032 012 175 293 323 001 337 171
370/ 315 056 001 303 364 001 000 000
```

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY | CHECKED BY |
|---|---|
| DATE | DATE |

DRAWING NO. SP2210-15 PAGE 3

modular micros ⇄ martin research

| KEY | FUNCTION |
|---|---|
| 6 | Display one of 64 keys. The six LED digits on the 420 Console board are divided into three pairs. Each pair displays the value (in hexadecimal) of the key pressed. If a shift key--*M* or *=K*--is depressed at the same time, it adds a corresponding amount to the value displayed. That is, with no shift keys depressed, the 16 hex values from 00 to 0F are displayed; with the *M* key depressed, the hex values from 10 to 1F are displayed; with the *=K* key pressed, hex values from 20 to 2F are displayed; and with both the *M* and the *=K* keys pressed, the hex values from 30 through 3F are displayed. |
| 7 | Go to PROM 2. That is, jump to location 002 000. |

To remove the computer from control by the *DEMO PROM*, it is necessary to provide it with an interrupt.

REV A

```
                        .
                        .       ONE PAGE MONITOR PROGRAM TO CONTROL KEYBOARD
                        .
                        .       INTEL80$                    . DEFINE 8080 ASSEMBLER STUFF
                        .
                        .       EQU'S FOR MONITOR PROGRAM
                        .
000                     STACK   EQU     014*256+CORE        . DEFINE STACK+1 FOR MONITOR POWER ON
000                     RAM     EQU     010*256+CORE        . RAM IS CURRENTLY PAGE 010
000                     PROM1   EQU     001*256+CORE        . NEXT PROM IS PAGE 001
                        .
110                     C8      EQU     841-1               . 8 MS TIME CONSTANT
273                     C6667   EQU     700-1               . 6.666 MS TIME CONSTANT
034                     C100    EQU     10525-1             . 100 MS TIME CONSTANT
                        .
315                     INP6    EQU     0315                . 8008 COMPATI BLE INP 006 INSTRUCTION
333                     OUT15   EQU     0333                . 8008 COMPATI BLE OUT 015 INSTRUCTION
335                     OUT16   EQU     0335                . 8008 COMPATI BLE OUT 016 INSTRUCTION
337                     OUT17   EQU     0337                . 8008 COMPATI BLE OUT 017 INSTRUCTION
                        .
                        .
                        .       SUBROUTINES USEFUL TO OTHER PROGRAMS:
                        .
                        .       LOC     NAME                DESCRIPTION
                        .       ---     ----                -----------
                        .
                        .       000     RST000              POWER ON REST (INITIALIZES STACK POINTER)
                        .       010     RST010              PROGRAM BREAKPOINT
                        .       020     RST020              TIMING LOOP TO DECREMENT B,C REGISTER
                        .       030     RST030              READ AND DEBOUNCE NEXT KEY
                        .       040     RST040              WAIT 1.0 SEC., OR UNTIL CALC KEY IS DEPRESSED
                        .       060     RST060              DEBOUNCE KEYBOARD
                        .       070     RST070              SEE IF ANY CALC KEY IS DEPRESSED(NO DEBOUNCE)
                        .       273     PARTHL              DISPLAY H AND L REGISTER IN OCTAL
                        .       334     PARTC               DISPLAY REGISTER C IN OCTAL (WITH FILLS)
                        .       362     EXP                 EXPLODE REG C INTO OCTAL DIGITS (FOR OUTPUT)
                        .
                        .
                        /.      MIKE3 - MONITOR FUNCTIONS
                        .       ------------------------
                        .
                        .       KEY     HEX                 FUNCTION
                        .       ---     ---                 --------
                        .
                        .       0-7     0-7                 INSERT DIGIT INTO DIPLAY ACCUM
                        .
                        .       8       8                   RECALL STACK POINTER (STATUS, REGS, USERS ..
                        .
                        .       9       9                   OFF TO PROM1
                        .
                        .       =       A                   WRITE DISPLAY ACCUM INTO MEMORY
                        .                                   CLEAR DISPLAY ACCUM
                        .                                   INCREMENT L AND H REGISTERS
                        .                                   DISPLAY CONTENTS OF MEMORY
                        .
                        .       -       8                   INCREMENT L AND H REGISTERS
                        .                                   CLEAR DISPLAY ACCUM
                        .                                   DISPLAY CONTENTS OF MEMORY
                        .
                        .       *       C                   LOAD H REGISTER FROM DISPLAY ACCUM
                        .
                        .       /       D                   LOAD L REGISTER FROM DISPLAY ACCUM
                        .                                   DISPLAY CONTENTS OF MEMORY
                        .
                        .       .       E                   OFF TO RAM
                        .
                        .       C       F                   CONTINUE WHERE INTERRUPTED
                        .                                   (WITH UPDATED REGISTERS IF CHANGED)
                        .
                        .   F1  0       10                  POWER ON RESET (INITIALIZE STACK POINTER)
                        .
                        .   F1  1       11                  OFF TO PROGRAM AT H,L
                        .
                        .   F1  2       12                  LOAD STACK POINTER FROM HL
                        .
                        .   F1  8       18                  DECREMENT L AND H REGISTER BY ONE
```

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY R | CHECKED BY D.M | DRAWING NO. SP2220-17 (PAGE 2) |
|---|---|---|
| DATE 12-10-75 | DATE 12-10-75 | |

EF4A

modular micros martin research

REV: A, B, C, D

```
                        .
                        .       HERE TO WAIT FOR NEXT FUNCTION
                        .
000171  367             NFNC    RST     6               . DEBOUNCE PREVIOUS KEY
000172  377             NFNC1   RST     7               . ANY KEYS DEPRESSED?
000173  372                     JM      NFNC1           . NOPE, KEEP LOOKING FOR DEPRESSED KEY
000174  172
000175  000
000176  303                     JMP     FNC             . YES, HANDLE NEXT FUNCTION
000177  146
000200  000


                        .
                        .       HERE TO HANDLE THE SPECIAL FUNCTIONS
                        .
000201  376             SFNC    CPI     033             . SPECIAL CASE OF SHIFT B (DEC H,L) ?
000202  033
000203  302                     JNZ     SFNC1           . SKIP IF NOT
000204  210
000205  000
000206  076                     MVI     A,023           . FORCE TO END OF JUMP TABLE
000207  023
000210  376             SFNC1   CPI     024             . WITHIN RANGE OF OUR FUNCTIONS ?
000211  024
000212  322                     JNC     (B) DISMEM      . NOPE, LOOK FOR NEXT FUNCTION
(B) 000213  125
000214  000
000215  006                     MVI     B,H(FTAB)       . LOAD PAGE ADDRESS OF JUMP TABLE
000216  000
000217  306                     ADI     L(FTAB-8)       . GENERATE OFFSET INTO TABLE
000220  217
000221  117                     MOV     C,A             . INTO PROPER REGISTER FOR MEMORY REFERENCE
000222  012                     LDAX    B               . FETCH ROUTINE OFFSET
000223  117                     MOV     C,A             . INTO PROPER REGISTER FOR MEMORY REFERENCE
000224  305                     PUSH    B               . SAVE ROUTINE ADDRESS ON STACK (FOR RET)
000225  367                     RST     6               . DEBOUNCE KEY
000226  311                     RET                     . OFF TO THAT ROUTINE
                        .
                        .       TABLE OF ROUTINE OFFSETS
                        .
000227  252             FTAB    +       L(PGM8)         . 8 = RESTORE STACK POINTER (SP)
000230  377                     +       L(PGM9)         . 9 = OFF TO PROM1
000231  113                     +       L(PGMA)         . A = WRITE MEMORY
000232  114                     +       L(PGMB)         . B = READ MEMORY
000233  120                     +       L(PGMC)         . C = LOAD H REGISTER
000234  122                     +       L(PGMD)         . D = LOAD L REGISTER
000235  262                     +       L(PGME)         . E = OFF TO PROGRAM IN RAM
000236  265                     +       L(PGMF)         . F = CONTINUE AFTER LAST INTERRUPT
000237  000                     +       L(RST000)       . 10 = POWER ON (SETUP MONITOR STACK)
000240  272                     +       L(PGM11)        . 11 = OFF TO PROGRAM IN RAM AT H,L
(C) 000241  256                 +       L(PGM12)        (C) . 12 = LOAD STACK POINTER FROM H L
000242  116                     +       L(PGM18)        . 18 = DECH,L AND THEN READ MEMORY
                        .
                        .       TEST FOR DEBUG PROM
(D) 000243  072         PGMM    LDA     (D) 001374      . TEST
(D) (C) 000244  374                                     . FOR
(D) 000245  001                                         . DEBUG
000246  267                     (D) ORA     A           . PROM
000247  312                     JZ      001375          .
000250  375                                             .
000251  001
```

(B) REV. WAS NFNC

(A) REV. 000213 171

(B) REV. 000 241 251

(B) REV. READ

```
000 243  000  NOP · FILL
000 244  000  NOP · FILL
000 245  000  NOP · FILL
000 246  000  NOP · FILL
000 247  000  NOP · FILL
000 248  000  NOP · FILL
```

(C) REV. READ

```
000 243  072  LDA 001377
000 244  377
000 246  075  DCR
```

252 041
252 000
[illegible] 000
[illegible] 071
[illegible]
[illegible]
[illegible]

(D) NOTES ON CHANGES:

LOC. 000 213 -- ILLEGAL CHARACTER RETURNS TO BLINKING STATE

LOC. 000 243 -- PROVISION FOR DEBUG PROM

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R | CHECKED BY D.M | DRAWING NO. |
|---|---|---|---|
| | DATE 12-10-75 | DATE 12-10-75 | SP 2220-17 (PAGE 6) |

EF4A

| REV |
|---|
| A |
| B |
| C |
| D |

```
;       REGISTER AND STATUS SAVEAREA:
;       -----------------------------
;
;       PAGE,OFFSET             REGISTER
;       -----------             --------
;
;       SP                      L
;       SP+1                    H
;       SP+2                    E
;       SP+3                    D
;       SP+4                    C
;       SP+5                    B
;       SP+6                    PSW
;       SP+7                    A
;       SP+10                   OLD PC (LSB)
;       SP+11                   OLD PC (MSB)
```

```
;       MIKE3 MONITOR
;       -------------
;
;       HERE ON POWER ON (MASTER RESET), SETUP MONITOR STACK
;
000000  061     RST000  LXI     SP,STACK        ; SETUP STACK POINTER
000001  000
000002  014
(B) 000003  373         EI                      ; FILL
000004  000             NOP                     ; FILL
000005  303             JMP     PGM8            ; OFF TO DISPLAY STACK POINTER
000006  252
000007  000

;       INTERRUPT OR PROGRAM BREAKPOINT COMES HERE
;       SAVE STATUS AND OFF TO KEYBOARD MONITOR PROGRAM
;
000010  365     X       RST010  PUSH    PSW             ; SAVE PSW AND A REGISTER
000011  305             PUSH    B               ; SAVE B AND C REGISTERS
000012  325             PUSH    D               ; SAVE D AND E REGISTERS
000013  345             PUSH    H               ; SAVE H AND L REGISTERS
000014  373             EI                      ; ENABLE INTERRUPTS
000015  303             JMP     PGMM (B)        ; OFF TO START THE HLINKING MONITOR
(C) 000016  243
000017  000

;       TIMING LOOP: B,C CONTAINS LOOP COUNTER
;       LOOP COUNT EQUALS N =       (USEC-8.5)/9.5 - 1
;       WAITS FROM 18.0 USEC TO 620 MS AVAILABLE
;       DESTROYS A,B,C
;
000020  076     RST020  MVI     A,0377          ; SETUP TERMINATION CONDITION
000021  377
000022  013     LOOP    DCX     B               ; (B,C) = (B,C) - 1
000023  270             CMP     B               ; HAVE WE REACHED THE END?
000024  302             JNZ     LOOP            ; NOPE --> LOOP UNTIL WE DO
000025  022
000026  000
000027  311             RET                     ; RETURN

;       HERE TO READ AND DEBOUNCE NEXT KEY
;       REGISTER C CONTAINS KEY NUMBER
;       DESTROYS A,B,C,D
;
000030  377     X>      RST030  RST     7               ; ANY KEYS DEPRESSED?
000031  372             JM      RST030          ; LOOP UNTIL A KEY IS DEPRESSED
000032  030
000033  000
                    (D)
(B) 000034  305         PUSH    B               ; COPY DEPRESSED KEY NUMBER
000035  367             RST     6               ; DEBOUNCE KEY
                    (D)
(B) 000036  301         POP     B               ; COPY DEPRESSED KEY NUMBER
000037  311             RET                     ; RETURN
```

(A) REV. 000 003 000 NOP

(A) REV. PGM 8
(B) WAS 000 016 252

(A) REV. 000 034 MOV D,C
(A) REV. 000 036 MOV C,D

(C) REV. 000 034 PUSH D
(C) REV. 000 036 POP D

E: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY R | CHECKED BY D.M | DRAWING NO. SP2220-17 (PAGE 3) |
|---|---|---|
| DATE 12-10-75 | DATE 12-10-75 | |

EF4A

modular micros martin research

REV
A
B

```
                        .
                        .       HERE TO APPEAR TO PAUSE TO OPERATOR -->
                        .       UNLESS A KEY IS DEPRESSED
                        .
                        .       CONDITION CODE SIGN = 1 IF NO KEYS DEPRESSED
                        .       CONDITION CODE SIGN = 0 IF KEYS DEPRESSED
                        .       REGISTER C CONTAINS DEPRESSED KEY NUMBER
                        .       DESTROYS REGISTERS A,B,C
                        .
                        .       NOTE:   KEYS ARE NOT DEBOUNCED (USE RST030 FOR DEBOUNCED READ)
                        .
000040  325     RST040  PUSH    D               . SAVE D,E REGISTERS
000041  036             MVI     E,0200          . OUT SIDE LOOP COUNTER
000042  200
                        .
                        .       INNER LOOP OF A FEW MILLISECONDS (10 MS.)
                        .
000043  035     RST043  DCR     E               . ANY MORE TIMES THRU THE LOOP ?
000044  372             JM      RST056          . NOPE --> POP D, E RETURN
000045  056
000046  000
000047  006             MVI     B,H(C8)-1       . MSB OF TIME CONSTANT OF 8 MS (SO IT FITS)
000050  002
000051  327             RST     2               . WAIT 10 MS.
000052  377             RST     7               . ANY KEYS DEPRESSED ?
000053  372             JM      RST043          . NOPE, GO ON LIKE THIS FOR 1.0 SECONDS
000054  043
000055  000
000056  321     RST056  POP     D               . RESTORE D AND E REGISTER
000057  311             RET                     . RETURN
                        .
                        .       HERE TO DEBOUNCE KEYS
                        .       DESTROYS REGISTERS A,B,C
                        .
000060  006     RST060  MVI     B,H (C100)-1    . MSB OF TIME CONSTANT  (SO IT FITS)
000061  020                                                              (A) REV. 000061 050
000062  327             RST     2               . WAIT 100 MS BEFORE TESTING KEYS AGAIN
000063  377             RST     7               . ANY KEYS DEPRESSED ?
000064  362             JP      RST060          . LOOP UNTIL NO KEYS DEPRESSED
000065  060
000066  000
000067  311             RET                     . RETURN
                        .
                        .       HERE TO SEE IF ANY KEYS DEPRESSED ON KEYBOARD
                        .       CONDITION CODE SIGN = 1 NO KEYS DEPRESSED
                        .       CONDITION CODE SIGN = 0 KEYS DEPRESSED
                        .       REGISTER C CONTAINS KEY NUMBER
                        .       DESTROYS REGISTER A,B,C
                        .
000070  001     RST070  LXI     B,INP6*256+020  . INP 006 INSTRUCT, NUMBER OF KEYS TO POLL
000071  020
000072  315
                        .
                        .       LOOP TO CHECK NEXT KEY (MUST POLL EACH OF 15 KEYS)
                        .
000073  015     RST073  DCR     C               . DECREMENT KEY LOOP COUNTER
000074  370             RM                      . NO MORE KEYS TO POLL, RETURN SIGN=1
000075  012             LDAX    B               . READ FROM KEYBOARD TO SEE IF KEY DEPRESSED
000076  017             RRC                     . ISOLATE LOW ORDER BIT  (KEY DEPRESSED FLAG)
000077  322             JNC     RST073          . THIS KEY NOT DEPRESSED, CHECK NEXT KEY
000100  073
000101  000
                        .
                        .       KEY IS DEPRESSED, NOW HANDLE SHIFT KEYS (FOUR FUNCTIONS POSSIBLE)
                        .
                        .       VALUE   FUNCTION        SHIFT KEYS DEPRESSED
                        .       -----   --------        --------------------
                        .
                        .       00-15   F0              NO SHIFT KEYS
                        .       16-31   F1               M SHIFT KEY
                        .       32-47   F2              =K SHIFT KEY
                        .       48-63   F3              M AND =K SHIFT KEYS (BOTH)
                        .
                        .       NOTE:   REG A = 011111KM (WHERE KM ARE THE SHIFT KEYS)
                        .
000102  346             ANI     003             . ISOLATE THE KM SHIFT KEYS
000103  003
000104  007             RLC                     . MULTIPLY THE
000105  007             RLC                     . SHIFT KEY FUNCTIONS
000106  007             RLC                     . F0,F1,F2,F3 BY 16
000107  007             RLC                     . BY SHIFTING LEFT 4 BITS
000110  201             ADD     C               . ADD IN KEY  (NOW SIGN =0)
000111  117             MOV     C,A             . COPY FOR OTHERS
000112  311             RET                     . RETURN
```

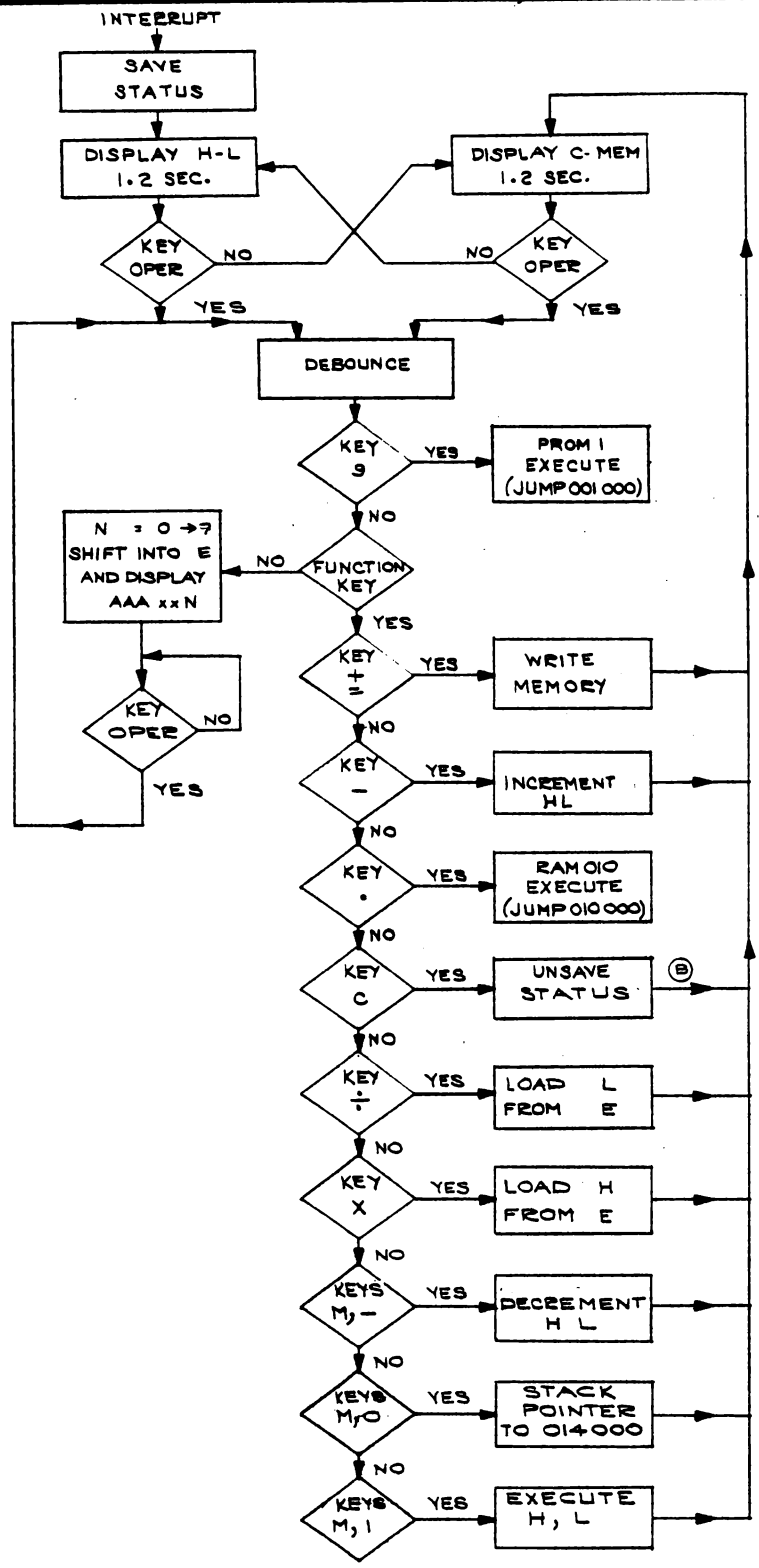
*NOTE: REV. B IMPROVES KEYBOARD RESPONSE*

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R | CHECKED BY D.M. | DRAWING NO. SP 2220-17 (PAGE 4) |
|---|---|---|---|
| | DATE 12-10-75 | DATE 12-10-75 | |

EF4A

modular micros martin research

# MIKE 3
# MONITOR FLOW CHART




| REV |
|---|
| A |
| B |
| |

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY | CHECKED BY | DRAWING NO. |
|---|---|---|
| DATE 12-10-75 | DATE 12-10-75 | SP 2220-17 (PAGE 1 of 8) |

F4A

modular micros
martin research

| REV |
|---|
| A |
| B |
| |

(B) REV. REMOVED
000 251 371 PGM12
SPHL - LOAD STACK POINTER FROM H,L (MERGE)

```
                        ;
                        ;       8 = RECALL STACK POINTER (MERGES HERE WITH 0=0)
                        ;
000252 041      PGM8    LXI     H,0             . H+L=0
000253 000
000254 000
000255 071              DAD     SP              . HL=SP
(B) 000256 371  PGM12   SPHL                    . LOAD STACK POINTER FROM H,L (MERGE)      (A) REV. 000 256 373 EI
000257 303              JMP     XPGM            . DISPLAY NEW/OLD H,L (SP,SP+1)            . ENABLE INTERRUPTS
000260 123
000261 000
                        ;
                        ;       E = JUMP TO PROGRAM IN RAM
                        ;
000262 303      PGME    JMP     RAM             . OFF TO PROGRAM IN PAGE 010
000263 000
000264 010
                        ;
                        ;       F = CONTINUE AFTER INTERRUPT
                        ;
000265 341      PGMF    POP     H               . RESTORE H AND L REGISTERS
000266 321              POP     D               . RESTORE D AND E REGISTERS
000267 301              POP     B               . RESTORE B AND C REGISTERS
000270 361              POP     PSW             . RESTORE PSW AND A REGISTERS
000271 311              RET                     . RETURN
                        ;
                        ;       11 = OFF TO PROGRAM IN RAM DEFINED BY H,L
                        ;
000272 351      PGM11   PCHL                    . OFF TO PROGRAM AT H,L


                        ;
                        ;       HERE TO OUTPUT 'H' AND 'L' REGISTER TO HEX DISPLAY PORTS
                        ;
                        ;       DESTROYS A,B,C
                        ;
000273 325      PARTHL  PUSH    D               . SAVE D AND E
000274 115              MOV     C,L             . LOAD LEAST SIGNIFICANT WORD
000275 315              CALL    EXP             . EXPLODE OCTAL DIGITS
000276 362
000277 000
000300 026              MVI     D,OUT15         . SETUP FOR 8008 COMPATI BLE I/O
000301 333
000302 137              MOV     E,A             . COPY DATA INTO PROPER REGISTER
000303 032              LDAX    D               . OUTPUT RIGHT 2 DIGITS
000304 174              MOV     A,H             . LOAD MOST SIGNIFICANT WORD
000305 017              RRC                     . MULTIPLY
000306 017              RRC                     . BY 2**3
000307 017              RRC                     . TO MOVE 3 BITS
000310 117              MOV     C,A             . SAVE FOR H REGISTER
000311 017              RRC                     . ONE MORE BIT
000312 346              ANI     0160            . EXTRACT -210----
000313 160
000314 260              ORA     B               . FORM    -210--76
000315 026              MVI     D,OUT16         . SETUP FOR 8008 COMPATI BLE I/O
000316 335
000317 137              MOV     E,A             . COPY DATA INTO PROPER REGISTER
000320 032              LDAX    D               . OUTPUT ONE H DIGIT AND ONE L DIGIT
000321 315              CALL    EXP             . EXPLODE MODIFIED H REGISTER
000322 362
000323 000
000324 346              ANI     077             . KILL UNUSED HIGH ORDER BITS
000325 077
000326 026              MVI     D,OUT17         . SETUP FOR 8008 COMPATI BLE I/O
000327 337
000330 137              MOV     E,A             . COPY DATA INTO PROPER REGISTER
000331 032              LDAX    D               . OUTPUT THE HIGH TWO H DIGITS
000332 321              POP     D               . RESTORE D AND E REGISTER
000333 311              RET                     . RETURN
```

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY | CHECKED BY | DRAWING NO. |
|---|---|---|
| DATE 12-10-75 | DATE 12-10-75 | SP2220-17 (PAGE 7) |

EF4A

modular micros martin research

| REV |
|---|
| A |
| |
| |

```
                        ;
                        ;       HERE TO OUTPUT BINARY REGISTER 'C' AND FILL CHARS IN REG 'D'
                        ;       DESTORYS REGISTER A,B
                        ;
000334  345            PARTC   PUSH    H               . SAVE H AND L
000335  315                    CALL    EXP             . EXPLODE INTO OCTAL DIGITS
000336  362
000337  000
000340  046                    MVI     H,OUT15         . SETUP FOR 8008 COMPATI BLE I/O INSTRUCTIONS
000341  333
000342  157                    MOV     L,A             . COPY DATA INTO PROPER REGISTER
000343  176                    MOV     A,M             . OUTPUT RIGHTMOST TWO DIGITS
000344  172                    MOV     A,D             . COPY FILL CHARACTER
000345  346                    ANI     0360            . CLEAR RIGHT DIGIT
000346  360
000347  260                    ORA     B               . FILL CHARACTER, LAST OCTAL DIGIT
000350  046                    MVI     H,OUT16         . 8008 COMPATI BLE OUT 016 INSTRUCTION
000351  335
000352  157                    MOV     L,A             . COPY DATA INTO PROPER REGISTER
000353  176                    MOV     A,M             . OUTPUT IT
000354  046                    MVI     H,OUT17         . 8008 COMPATI BLE OUT 017 INSTRUCTION
000355  337
000356  152                    MOV     L,D             . COPY DATA INTO PROPER REGISTER
000357  176                    MOV     A,M             . OUTPUT IT
000360  341                    POP     H               . RESTORE H AND L REGISTER
000361  311                    RET                     . RETURN
                        ;
                        ;       EXPLODE BINARY REG C INTO PRINTABLE OCTAL FOR HEX OUTPUT PORTS
                        ;
                        ;       REGISTER A CONTAINS:    -543-210
                        ;       REGISTER B CONTAINS:    ------76
                        ;
                        ;       NOTE THAT '-' DENOTES ZERO BITS
                        ;
000362  171            EXP     MOV     A,C             . COPY BINARY REGISTER
000363  007                    RLC                     . POSITION BITS
000364  007                    RLC                     . 7 AND 6
000365  346                    ANI     003             . FORM ------76
000366  003
000367  107                    MOV     B,A             . SAVE ------76
000370  171                    MOV     A,C             . COPY BINARY REGISTER TO CONVERT
000371  346                    ANI     0370            . ISOLATE XX543---
000372  370
000373  201                    ADD     C               . FORM    X543-210
000374  346                    ANI     0177            . FORM    -543-210
000375  177
000376  311                    RET                     . RETURN
                        ;
                        ;       9 = OFF TO PROM1
                        ;
000377  000            PGM9    NOP                     . FALL THRU TO NEXT PROM
                        ;
                                END
```

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R | CHECKED BY D.M | DRAWING NO. SP 2220-17 (PAGE 8) |
|---|---|---|---|
| | DATE 12-10-75 | DATE 12-10-75 | |

EF4A

modular micros martin research

REV A

```
                        ;
                        ;       ONE PAGE MONITOR PROGRAM TO CONTROL KEYBOARD
                        ;
                        ;       INTEL80$                    . DEFINE 8080 ASSEMBLER STUFF
                        ;
                        ;       EQU'S FOR MONITOR PROGRAM
                        ;
000                     STACK   EQU     014*256+CORE        . DEFINE STACK+1 FOR MONITOR POWER ON
000                     RAM     EQU     010*256+CORE        . RAM IS CURRENTLY PAGE 010
000                     PROM1   EQU     001*256+CORE        . NEXT PROM IS PAGE 001
                        ;
110                     C8      EQU     841-1               . 8 MS TIME CONSTANT
273                     C6667   EQU     700-1               . 6.666 MS TIME CONSTANT
034                     C100    EQU     10525-1             . 100 MS TIME CONSTANT
                        ;
315                     INP6    EQU     0315                . 8008 COMPATIBLE INP 006 INSTRUCTION
333                     OUT15   EQU     0333                . 8008 COMPATIBLE OUT 015 INSTRUCTION
335                     OUT16   EQU     0335                . 8008 COMPATIBLE OUT 016 INSTRUCTION
337                     OUT17   EQU     0337                . 8008 COMPATIBLE OUT 017 INSTRUCTION
                        ;
                        ;
                        ;       SUBROUTINES USEFUL TO OTHER PROGRAMS:
                        ;
                        ;       LOC     NAME            DESCRIPTION
                        ;       ---     ----            -----------
                        ;
                        ;       000     RST000          POWER ON REST (INITIALIZES STACK POINTER)
                        ;       010     RST010          PROGRAM BREAKPOINT
                        ;       020     RST020          TIMING LOOP TO DECREMENT B,C REGISTER
                        ;       030     RST030          READ AND DEBOUNCE NEXT KEY
                        ;       040     RST040          WAIT 1.0 SEC., OR UNTIL CALC KEY IS DEPRESSED
                        ;       060     RST060          DEBOUNCE KEYBOARD
                        ;       070     RST070          SEE IF ANY CALC KEY IS DEPRESSED(NO DEBOUNCE)
                        ;       264     PARTHL          DISPLAY H,L REGS, CONTENTS OF MEMORY IN HEX
                        ;       305     PARTC           DISPLAY REGISTER C IN HEX
                        ;       306     PARTA           DISPLAY REGISTER A IN HEX
                        ;       324     EXP             EXPLODE BINARY REG C INTO OCTAL PARTS FOR HEX
                        ;
                        ;
                        /.      MIKE3 - MONITOR FUNCTIONS
                        ;       ------------------------
                        ;
                        ;       KEY     HEX             FUNCTION
                        ;       ---     ---             --------
                        ;
                        ;       0-F     0-F             INSERT HEX DIGIT INTO DISPLAY ACCUM
                        ;
                        ;  F1   8       18              RECAL STACK POINTER
                        ;
                        ;  F1   9       19              OFF TO PROM1
                        ;
                        ;  F1   =       1A              WRITE DISPLAY ACCUM INTO MEMORY
                        ;                               CLEAR DISPLAY ACCUM
                        ;                               INCREMENT L AND H REGISTERS
                        ;                               DISPLAY CONTENTS OF MEMORY
                        ;
                        ;  F1   -       1B              INCREMENT L AND H REGISTER
                        ;                               CLEAR DISPLAY ACCUM
                        ;                               DISPLAY CONTENTS OF MEMORY
                        ;
                        ;  F1   *       1C              LOAD H REGISTER FROM DISPLAY ACCUM
                        ;
                        ;  F1   /       1D              LOAD L REGISTER FROM DISPLAY ACCUM
                        ;                               DISPLAY CONTENTS OF MEMORY
                        ;
                        ;  F1   .       1E              OFF TO RAM PAGE 010
                        ;
                        ;  F1   C       1F              CONTINUE WHERE INTERRUPTED
                        ;                               (WITH UPDATED REGISTERS IF CHANGED)
                        ;
                        ;  F1   0       10              POWER ON RESET (INITIALIZE STACK POINTER)
                        ;
                        ;  F1   1       11              OFF TO PROGRAM AT H,L
                        ;
                        ;  F1   2       12              LOAD STACK POINTER FROM H L
                        ;
                        ;  F2   8       28              DECREMENT L AND H REGISTER BY ONE
                        ;  F1   6       16              DECREMENT L AND H REGISTER BY ONE
```

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY R | CHECKED BY DM | DRAWING NO. SP 2220-16 (PAGE 1 of 6) |
|---|---|---|
| DATE 12-10-75 | DATE 12-10-75 | |

EF4A

modular micros martin research

REV
A
B
C

```
                        ;       REGISTER AND STATUS SAVEAREA:
                        ;       ----------------------------
                        ;
                        ;       PAGE,OFFSET             REGISTER
                        ;       -----------             --------
                        ;
                        ;       SP                      L
                        ;       SP+1                    H
                        ;       SP+2                    E
                        ;       SP+3                    D
                        ;       SP+4                    C
                        ;       SP+5                    B
                        ;       SP+6                    PSW
                        ;       SP+7                    A
                        ;       SP+10                   OLD PC (LSB)
                        ;       SP+11                   OLD PC (MSB)



                        ;.      MIKE3 MONITOR
                        ;       -------------
                        ;
                        ;       HERE ON POWER ON (MASTER RESET), SETUP MONITOR STACK
                        ;
000000  061             RST000  LXI     SP,STACK        . SETUP STACK POINTER
000001  000
000002  014
(B) 000003  373                 EI                      . FILL                          (A) REV.
000004  000                     NOP                     . FILL                          000 003  000  NOP
000005  303                     JMP     PGM8            . OFF TO DISPLAY STACK POINTER
000006  237
000007  000
                        ;
                        ;       INTERRUPT OR PROGRAM BREAKPOINT COMES HERE
                        ;       SAVE STATUS AND OFF TO KEYBOARD MONITOR PROGRAM
                        ;
000010  365             RST010  PUSH    PSW             . SAVE PSW AND A REGISTER
000011  305                     PUSH    B               . SAVE B AND C REGISTERS
000012  325                     PUSH    D               . SAVE D AND E REGISTERS
000013  345                     PUSH    H               . SAVE H AND L REGISTERS
000014  373                     EI                      . ENABLE INTERRUPTS
000015  303                     JMP     PGM8            . OFF TO START THE BLINKING MONITOR
000016  237
000017  000
                        ;
                        ;       TIMING LOOP: B,C CONTAINS LOOP COUNTER
                        ;       LOOP COUNT EQUALS N =   (USEC-8.5)/9.5 - 1
                        ;       WAITS FROM 18.0 USEC TO 620 MS AVAILABLE
                        ;       DESTROYS A,B,C
                        ;
000020  076             RST020  MVI     A,0377          . SETUP TERMINATION CONDITION
000021  377
000022  013             LOOP    DCX     B               . (B,C) = (B,C) - 1
000023  270                     CMP     B               . HAVE WE REACHED THE END?
000024  302                     JNZ     LOOP            . NOPE --> LOOP UNTIL WE DO
000025  022
000026  000
000027  311                     RET                     . RETURN
                        ;
                        ;       HERE TO READ AND DEBOUNCE NEXT KEY
                        ;       REGISTER C CONTAINS KEY NUMBER
                        ;       DESTROYS A,B,C,D
                        ;
000030  377             RST030  RST     7               . ANY KEYS DEPRESSED?
000031  372                     JM      RST030          . LOOP UNTIL A KEY IS DEPRESSED
000032  030
000033  000                 (C) REV.
(B) 000034  305                 PUSH B                  . COPY DEPRESSED KEY NUMBER     (A) REV.
000035  367                     RST     6               . DEBOUNCE KEY                  000 034  121  MOV D,C
(B) 000036  301             (C) REV.    POP B           . COPY DEPRESSED KEY NUMBER     (A) REV.
000037  311                     RET                     . RETURN                        000 036  112  MOV C,D

                                                                                        (B) REV.
                                                                                        000 034  PUSH D

                                                                                        (B) REV.
                                                                                        000 036  POP D
```

| 1 | 2 | 3 |
|---|---|---|
| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R / CHECKED BY DM | DRAWING NO. SP2220-16 (PAGE 2) |
| | DATE 12-10-75 / DATE 12-10-75 | |

EF4A

modular micros martin research

REV A

```
                                ;
                                ;       HERE TO HANDLE THE SPECIAL FUNCTIONS
                                ;
000167  376             SFNC    CPI     053             . SPECIAL CASE OF F2 B (DEC H,L) ?
000170  053
000171  302                     JNZ     SFNC1           . SKIP IF NOT
000172  176
000173  000
000174  076                     MVI     A,026           . SAME AS F1 6
000175  026
000176  376             SFNC1   CPI     040             . WITHIN RANGE OF OUR SPECIAL FUNCTIONS ?
000177  040
000200  322                     JNC     NFNC            . NOPE, LOOK FOR NEXT FUNCTION
000201  157
000202  000
000203  006                     MVI     B,H(FTAB)       . LOAD PAGE ADDRESS OF JUMP TABLE
000204  000
000205  306                     ADI     L(FTAB-020)     . GENERATE OFFSET INTO TABLE
000206  175
000207  117                     MOV     C,A             . INTO PROPER REGISTER FOR MEMORY REFERENCE
000210  012                     LDAX    B               . FETCH ROUTINE OFFSET
000211  117                     MOV     C,A             . INTO PROPER REGISTER FOR MEMORY REFERENCE
000212  305                     PUSH    B               . SAVE ROUTINE ADDRESS ON STACK (FOR RET)
000213  367                     RST     6               . DEBOUNCE KEY
000214  311                     RET                     . OFF TO THAT ROUTINE
                                ;
                                ;       TABLE OF  ROUTINE OFFSETS
                                ;
000215  000             FTAB    +       L(RST000)       . 10 = POWER ON (RESET STACK POINTER)
000216  260                     +       L(PGM11)        . 11 = OFF TO RAM AT H,L ADDRESS
000217  235                     +       L(PGM12)        . 12 = LOAD STACK POINTER FROM H L
000220  157                     +       L(PGM13)        . 13 =
000221  157                     +       L(PGM14)        . 14 =
000222  157                     +       L(PGM15)        . 15 =
000223  116                     +       L(PGM16)        . 16 = DEC H,L BY ONE (SAME AS 28)
000224  157                     +       L(PGM17)        . 17 =
000225  237                     +       L(PGM8)         . 18 = RESTORE STACK POINTER
000226  261                     +       L(PGM9)         . 19 = OFF TO NEXT PROM
000227  113                     +       L(PGMA)         . 1A = WRITE MEMORY FROM DISPLAY ACCUM
000230  114                     +       L(PGMB)         . 1B = READ NEXT MEMORY LOCATION
000231  120                     +       L(PGMC)         . 1C = LOAD H REGISTER FROM DISPLAY ACCUM
000232  122                     +       L(PGMD)         . 1D = LOAD L REGISTER FROM DISPLAY ACCUM
000233  250                     +       L(PGME)         . 1E = OFF TO PROGRAM IN PAGE 010 LOC 000
000234  253                     +       L(PGMF)         . 1F = CONTINUE AFTER LAST INTERRUPT

                                ;
                                ;       12 = REMOVE STACK ENTRIES FOR THIS INTERRUPT LEVEL
                                ;
000235  371             PGM12   SPHL                    . LOAD STACK POINTER FROM HL (MERGE)
000236  000                     NOP                     . NOT USED
                                ;
                                ;       8 = RECALL STACK POINTER (MERGES HERE WITH D=0)
                                ;
000237  041             PGM8    LXI     H,0             .
000240  000
000241  000                                             .
000242  071                     DAD     SP              . H,L=SP
000243  000                     NOP                     . FILL
000244  373                     EI                      . ENABLE INTERRUPTS
000245  303                     JMP     XPGM            . DISPLAY NEW/OLD H,L (SP,SP+1)
000246  123
000247  000
                                ;
                                ;       E = JUMP TO PROGRAM IN RAM
                                ;
000250  303             PGME    JMP     RAM             . OFF TO PROGRAM IN PAGE 010
000251  000
000252  010
                                ;
                                ;       F = CONTINUE AFTER INTERRUPT
                                ;
000253  341             PGMF    POP     H               . RESTORE H AND L REGISTERS
000254  321                     POP     D               . RESTORE D AND E REGISTERS
000255  301                     POP     B               . RESTORE B AND C REGISTERS
000256  361                     POP     PSW             . RESTORE PSW AND A REGISTERS
000257  311                     RET                     . RETURN
                                ;
                                ;       11 = OFF TO PROGRAM IN RAM DEFINED BY H,L
                                ;
000260  351             PGM11   PCHL                    . OFF TO PROGRAM AT H,L
                                ;
                                ;       9 = OFF TO PROM1
                                ;
000261  303             PGM9    JMP     PROM1           . OFF TO THE NEXT PROM
000262  000
000263  001
```

| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R | CHECKED BY D.M | DRAWING NO. SP2220-16 (PAGE 5) |
|---|---|---|---|
| | DATE 12-10-75 | DATE 12-10-75 | |

EF4A

modular micros martin research

REV
A

```
                                ;
                                ;       A = WRITE NEXT LOCATION , THEN INC H AND L REGISTER
                                ;
000113  163             PGMA    MOV     M,E             . WRITE MEMORY AND MERGE TO INC
                                ;
                                ;       B = INCREMENT H AND L REGISTER, THEN READ
                                ;
000114  043             PGMB    INX     H               . INCREMENT H AND L REGISTERS
000115  043                     INX     H               . INCREMENT H AND L (SAVES JUMP) FOR MERGE
                                ;
                                ;       1B = DECREMENT H AND L REGISTER, THEN READ
                                ;
000116  053             PGM1B   DCX     H               . DECREMENT H AND L REGISTER
000117  134                     MOV     E,H             . COPY H FOR MERGE (SAVES JUMP)
                                ;
                                ;       C = LOAD THE 'H' REGISTER
                                ;
000120  143             PGMC    MOV     H,E             . LOAD THE H REGISTER FROM DISPLY ACCUM
000121  135                     MOV     E,L             . COPY L REGISTER FOR MERGE (SAVES JMP)
                                ;
                                ;       D = LOAD THE 'L' REG, DISPLAY NEXT MEMORY LOCATION
                                ;
000122  153             PGMD    MOV     L,E             . LOAD THE 'L' REGISTER FROM ACCUM DISPLAY
                                ;
                                ;       HERE TO CLEAR DISPLAY ACCUM (REGISTER E)
                                ;
000123  036             XPGM    MVI     E,0             . CLEAR DISPLAY ACCUM
000124  000
                                ;
                                ;       HERE TO DISPLAY CONTENTS OF MEMORY
                                ;
000125  315             DISMEM  CALL    PARTHL          . DISPLAY H,L AND CONTENTS OF MEMORY THERE
000126  264
000127  000
000130  347                     RST     4               . PAUSE 1.3 SECONDS, OR UNTIL
000131  372                     JM      DISMEM          . NO KEY DEPRESSED , REPEAT BLINKING
000132  125
000133  000
                                ;
                                ;       NOTE FALL THRU
                                ;
                                ;
                                ;       HERE TO GET NEXT FUNCTION
                                ;
000134  376             FNC     CPI     020             . HEX DIGITS 0-F ?
000135  020
000136  322                     JNC     SFNC            . NOPE SPECIAL FUNCTION
000137  167
000140  000
                                ;
                                ;       HERE TO INSERT NEXT HEX DIGIT INTO DISPLAY ACCUM (REG E)
                                ;
000141  173                     MOV     A,E             . COPY DISPLAY ACCUM
000142  007                     RLC                     . SHIFT OVER
000143  007                     RLC                     .  FOUR BITS
000144  007                     RLC                     .   FOR NEXT
000145  007                     RLC                     . HEX DIGIT
000146  346                     ANI     0360            . CLEAR RIGHTMOST 4 BITS FOR NEW DIGIT
000147  360
000150  201                     ADD     C               . ADD IN NEW HEX DIGIT
000151  137                     MOV     E,A             . STORE UPDATED DISPLAY ACCUM
000152  026                     MVI     D,0252          . FILL CHARACTER = 'AA' FOR ACCUM
000153  252
000154  315                     CALL    PARTA           . OUTPUT TO CONSOLE DISPLAY BOARD
000155  306
000156  000
                                ;
                                ;       HERE TO WAIT FOR NEXT FUNCTION
                                ;
                        PGM13                           . INVALID KEY (FOR NOW)
                        PGM14                           . INVALID KEY (FOR NOW)
                        PGM15                           . INVALID KEY (FOR NOW)
                        PGM17                           . INVALID KEY (FOR NOW)
000157  307             NFNC    RST     6               . DEBOUNCE PREVIOUS KEY
000160  377             NFNC1   RST     7               . ANY KEYS DEPRESSED?
000161  372                     JM      NFNC1           . NOPE, KEEP LOOKING FOR DEPRESSED KEY
000162  160
000163  000
000164  303                     JMP     FNC             . YES, HANDLE NEXT FUNCTION
000165  134
000166  000
```

NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION.

| DRAWN BY | CHECKED BY | DRAWING NO. |
|---|---|---|
| DATE 12-10-75 | DATE 12-10-75 | SP2220-16 (PAGE 4) |

EF4A

modular micros martin research

# 8080 INSTRUCTION SET

## REGISTER TRANSFERS

MOV B, H = MOVE TO THE B REG. THE CONTENTS OF THE H REGISTER.

A = 7 C = 1 E = 3 L = 5
B = 0 D = 2 H = 4 M = 6

| | | DESTINATION A | B | C | D | E | H | L | M |
|---|---|---|---|---|---|---|---|---|---|
| SOURCE | A | 177 | 107 | 117 | 127 | 137 | 147 | 157 | 167 |
| | B | 170 | 100 | 110 | 120 | 130 | 140 | 150 | 160 |
| | C | 171 | 101 | 111 | 121 | 131 | 141 | 151 | 161 |
| | D | 172 | 102 | 112 | 122 | 132 | 142 | 152 | 162 |
| | E | 173 | 103 | 113 | 123 | 133 | 143 | 153 | 163 |
| | H | 174 | 104 | 114 | 124 | 134 | 144 | 154 | 164 |
| | L | 175 | 105 | 115 | 125 | 135 | 145 | 155 | 165 |
| | M | 176 | 106 | 116 | 126 | 136 | 146 | 156 | — |
| MOVE IMMEDIATE | MVI r | 076 | 006 | 016 | 026 | 036 | 046 | 056 | 066 |

## DOUBLE-PRECISION TRANSFERS

| | | DESTINATION B, C | D, E | H, L | SP | PC |
|---|---|---|---|---|---|---|
| SOURCE | D, E | —— | —— | XCHG 353 | —— | —— |
| | H, L | —— | XCHG 353 | —— | SPHL 371 | PCHL 351 |
| | IMMED. | LXI B 001 | LXI D 021 | LXI H 041 | LXI SP 061 | —— |

## INDIRECT TRANSFERS

| BYTES | INSTR. | REG. | DATA | MEMORY ADDR. BY: |
|---|---|---|---|---|
| 1 | STAX B 002 | A REGISTER | → | B, C |
| | LDAX B 012 | | ← | |
| | STAX D 022 | | → | D, E |
| | LDAX D 032 | | ← | |
| | MOV M, r SEE ABOVE | INDEX REG. r | → | H, L |
| | MOV r, M SEE ABOVE | | ← | |
| | MVI M 066 | 2ND BYTE OF INST. | → | |
| 2 | PUSH B 305 | B, C | → | SP (Stack Pointer) PUSH DECR. SP BY 2 |
| | POP B 301 | | ← | |
| | PUSH D 325 | D, E | → | |
| | POP D 321 | | ← | |
| | PUSH H 345 | H, L | → | |
| | POP H 341 | | ← | POP INCR. SP BY 2 |
| | PUSH PSW 365 | A, PSW | → | |
| | POP PSW 361 | | ← | |
| | XTHL 343 | H, L | ↔ | SP |

## DIRECT LOAD & STORE

| BYTES | INSTR. | REG. | DATA | MEMORY ADDR. BY: |
|---|---|---|---|---|
| 1 | STA 062 | A | → | 2nd AND 3rd BYTES OF INSTR. |
| | LDA 072 | | ← | |
| 2 | SHLD 042 | H, L | → | |
| | LHLD 052 | | ← | |

## INPUT/ OUTPUT

| INPUT | OUTPUT |
|---|---|
| IN 333 | OUT 323 |
| 2ND BYTE = I/O ADDR. | |

| UNUSED | |
|---|---|
| 010 | 313 |
| 020 | 331 |
| 030 | 335 |
| 040 | 355 |
| 050 | 375 |
| 060 | — |
| 070 | — |

Copyright © 1976
MARTIN RESEARCH
ALL RIGHTS RESERVED

## PROGRAM CONTROL

| | UN-CONDI-TIONAL | IN-DI-RECT | ZERO FALSE | ZERO TRUE | CARRY FALSE | CARRY TRUE | PARITY FALSE | PARITY TRUE | SIGN FALSE | SIGN TRUE |
|---|---|---|---|---|---|---|---|---|---|---|
| JUMP | JMP 303 | PCHL 351 | JNZ 302 | JZ 312 | JNC 322 | JC 332 | JPO 342 | JPE 352 | JP 362 | JM 372 |
| CALL (PUSH PC; JUMP) | CALL 315 | —— | CNZ 304 | CZ 314 | CNC 324 | CC 334 | CPO 344 | CPE 354 | CP 364 | CM 374 |
| RETURN (POP PC) | RET 311 | —— | RNZ 300 | RZ 310 | RNC 320 | RC 330 | RPO 340 | RPE 350 | RP 360 | RM 370 |

| RESTART | RST 0 307 | RST 1 317 | RST 2 327 | RST 3 337 | RST 4 347 | RST 5 357 | RST 6 367 | RST 7 377 |
|---|---|---|---|---|---|---|---|---|

## BIT MANIPULATION

| | | A | B | C | D | E | H | L | M |
|---|---|---|---|---|---|---|---|---|---|
| INCREMENT REGISTER | INR r | 074 | 004 | 014 | 024 | 034 | 044 | 054 | 064 |
| DECREMENT REGISTER | DCR r | 075 | 005 | 015 | 025 | 035 | 045 | 055 | 065 |

| | B, C | D, E | H, L | SP |
|---|---|---|---|---|
| INCREMENT REGISTER PAIR | INX B 003 | INX D 023 | INX H 043 | INX SP 063 |
| DECREMENT REGISTER PAIR | DCX B 013 | DCX D 033 | DCX H 053 | DCX SP 073 |

| | | | |
|---|---|---|---|
| ROTATE A | LEFT | RLC | 007 |
| | RIGHT | RRC | 017 |
| | LEFT THRU CARRY | RAL | 027 |
| | RIGHT THRU CARRY | RAR | 037 |

| | | | | |
|---|---|---|---|---|
| DECIMAL ADJUST ACCUMULATOR | DAA 047 | | | |
| COMPLEMENT A REGISTER | CMA 057 | | | |
| SET / COMPLEMENT CARRY | SET | STC 067 | COMP. | CMC 077 |

## ALU INSTRUCTIONS

| | | A | B | C | D | E | H | L | M | IMMED. |
|---|---|---|---|---|---|---|---|---|---|---|
| ADD r TO A → A | ADD r | 207 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | ADI 306 |
| ADD r + C TO A → A | ADC r | 217 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | ACI 316 |
| SUB r FROM A → A | SUB r | 227 | 220 | 221 | 222 | 223 | 224 | 225 | 226 | SUI 326 |
| SUB (r + C) FROM A → A | SBB r | 237 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | SBI 336 |
| AND r WITH A → A | ANA r | 247 | 240 | 241 | 242 | 243 | 244 | 245 | 246 | ANI 346 |
| EXCL. OR r WITH A → A | XRA r | 257 | 250 | 251 | 252 | 253 | 254 | 255 | 256 | XRI 356 |
| OR r WITH A → A | ORA r | 267 | 260 | 261 | 262 | 263 | 264 | 265 | 266 | ORI 366 |
| COMPARE r WITH A | CMP r | 277 | 270 | 271 | 272 | 273 | 274 | 275 | 276 | CPI 376 |

| | B, C | D, E | H, L | SP |
|---|---|---|---|---|
| DOUBLE ADD $r_2$, $r_1$ TO H, L → H, L | DAD B 011 | DAD D 031 | DAD H 051 | DAD SP 071 |

## MACHINE CONTROL

| | | |
|---|---|---|
| NO OPERATION | NOP | 000 |
| HALT | HLT | 166 |
| DISABLE INTERRUPT | DI | 363 |
| ENABLE INTERRUPT | EI | 373 |

## FLAGS

| | C Z S P |
|---|---|
| ALU INST | ✓ ✓ ✓ ✓ |
| DOUBLE ADD | ✓ — — — |
| DEC. ADJUST | ✓ ✓ ✓ ✓ |
| ROTATE | ✓ - - - |
| INCR./DECR. | - ✓ ✓ ✓ |
| INX / DCX | - - - - |
| SET/COMP. CARRY | ✓ - - - |
| POP PSW | ✓ ✓ ✓ ✓ |

| REV |
|---|
| A |
| B |
| |

```
                        ;
                        ;       HERE TO OUTPUT 'H' AND 'L' REGISTER TO HEX DISPLAY PORTS
                        ;       ALONG WITH CONTENTS OF MEMORY AT THAT ADDRESS
                        ;
                        ;       DESTROYS NO REGISTERS
                        ;
000264  365    PARTHL   PUSH    PSW             . SAVE STATUS, AND A REGISTER
000265  325             PUSH    D               . SAVE D AND E REGISTERS
000266  026             MVI     D,OUT17         . 8008 COMPATI BLE OUT 017 INSTRUCTION
000267  337
000270  134             MOV     E,H             . COPY H INTO PROPER REG FOR 8008 I/O
000271  032             LDAX    D               . OUTPUT H REGISTER TO OUT 017
000272  026             MVI     D,OUT16         . 8008 COMPATI BLE OUT 016 INSTRUCTION
000273  335
000274  135             MOV     E,L             . COPY L INTO PROPER REG FOR 8008 I/O
000275  032             LDAX    D               . OUTPUT L REGISTER TO OUT 016
000276  026             MVI     D,OUT15         (B). 8008 COMPATI BLE OUT 015 INSTRUCTION   (A) REV. WAS 016 INSTRUCTION
000277  333
000300  136             MOV     E,M             . COPY MEMORY WORD INTO PROPER REG
000301  032             LDAX    D               . AND OUTPUT TO 8008 OUT 015 PORT
000302  321             POP     D               . RESTORE D AND E REGISTERS FROM STACK
000303  361             POP     PSW             . RESTORE PSW AND A REGISTER
000304  311             RET                     . RETURN

                        ;
                        ;       HERE TO OUTPUT REG A (OR REG C) WITH FILL CHAR IN REG D
                        ;       THIS ROUTINE OPERATES IN HEX, NOT OCTAL
                        ;       DESTROYS REGISTER A
                        ;
000305  171    PARTC    MOV     A,C             . COPY C INTO PROPER REGISTER FOR MERGE
000306  345    PARTA    PUSH    H               . SAVE H AND L REGISTER
000307  046             MVI     H,OUT15         . 8008 COMPATI BLE OUT 015 INSTRUCTION
000310  333
000311  157             MOV     L,A             . COPY HEX WORD TO OUTPUT
000312  176             MOV     A,M             . OUTPUT IT TO 8008 OUT 015
000313  046             MVI     H,OUT16         . 8008 COMPATI BLE OUT 016 INSTRUCTION
000314  335
000315  152             MOV     L,D             . COPY FILL CHARS
000316  176             MOV     A,M             . OUTPUT FILL TO 8008 OUT 016
000317  046             MVI     H,OUT17         . 8008 COMPATI BLE OUT 017 INSTRUCTION
000320  337
000321  176             MOV     A,M             . OUTPUT FILL TO 8008 OUT 017
000322  341             POP     H               . RESTORE H AND L REGISTER
000323  311             RET                     . RETURN
                        ;
                        ;       EXPLODE BINARY REG C INTO PRINTABLE OCTAL FOR HEX OUTPUT PORTS
                        ;
                        ;       REGISTER A CONTAINS:    -543-210
                        ;       REGISTER B CONTAINS:    ------76
                        ;
                        ;       NOTE THAT '-' DENOTES ZERO BITS
                        ;
000324  171    EXP      MOV     A,C             . COPY BINARY REGISTER
000325  007             RLC                     . POSITION BITS
000326  007             RLC                     . 7 AND 6
000327  346             ANI     003             . FORM ------76
000330  003
000331  107             MOV     B,A             . SAVE ------76
000332  171             MOV     A,C             . COPY BINARY REGISTER TO CONVERT
000333  346             ANI     0370            . ISOLATE XX543---
000334  370
000335  201             ADD     C               . FORM    X543-210
000336  346             ANI     0177            . FORM    -543-210
000337  177
000340  311             RET                     . RETURN

                        END
```

| 1 | 2 | 3 |
|---|---|---|
| NOTE: PROPRIETARY INFORMATION. NOT TO BE DISCLOSED WITHOUT EXPRESS WRITTEN PERMISSION. | DRAWN BY R / CHECKED BY D.M | DRAWING NO. SP2220-16 (PAGE 6) |
| | DATE 12-10-75 / DATE 12-10-75 | |

EF4A

modular micros martin research

# MICROCOMPUTER CODING FORM

PROGRAM:

BY:

DATE:

PAGE OF

| | LOCATION | | INSTRUCTION CODE | | | LABEL | OPERATION | OPERANDS AND REMARKS |
|---|---|---|---|---|---|---|---|---|
| | H | L | 1 | 2 | 3 | | | |
| | 1 2 3 | 4 5 6 | 7 8 9 | 10 11 12 | 13 14 15 | 16 17 18 19 20 | 21 22 23 24 | 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 50 51 52 53 54 55 56 57 58 59 |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | | | | | | | | |
| 15 | | | | | | | | |
| 16 | | | | | | | | |
| 17 | | | | | | | | |
| 18 | | | | | | | | |
| 19 | | | | | | | | |
| 20 | | | | | | | | |
| 21 | | | | | | | | |
| 22 | | | | | | | | |
| 23 | | | | | | | | |
| 24 | | | | | | | | |
| 25 | | | | | | | | |
| 26 | | | | | | | | |

EF8-4/76

REORDER IN PADS OF 50 - CAT. NO. SP7080-$2.00-MARTIN RESEARCH, NORTHBROOK, IL 60062